CRT-2000-10

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

A STRATEGIC FLOW MODEL OF TRAFFIC ASSIGNMENT IN CAPACITATED NETWORKS

by

Patrice Marcotte
Sang Nguyen
Alexandre Schoeb

February 2000

100-05365/

CRT-2000-10

*A STRATEGIC FLOW MODEL OF TRAFFIC ASSIGNMENT IN CAPACITATED NETWORKS**

by

Patrice Marcotte[1]
Sang Nguyen[1]
Alexandre Schoeb[1]

* *This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Fonds pour la formation de chercheurs et l'aide à la recherche (FCAR).*

[1] Département d'informatique et de recherche opérationnelle et Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, CANADA H3C 3J7

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2000

ABSTRACT

This work pleads for the use of the concept of strategies, and their network-theoretic representation as hyperpaths, for modelling network assignment problems. While this concept describes adequatly the behaviour of users in transit systems, we show that it can apply as well to networks where arc capacities are rigid. This opens up a whole new field of research and raises several questions, both from the theoretical and computational point of views. These are investigated in the paper.

Keywords: Equilibrium. Traffic Assignment. Hyperpath. Capacities. Variational inequalities.

RÉSUMÉ

Dans ce document, nous proposons une approche entièrement nouvelle pour l'affectation d'equilibre dans le transport urbain, basée sur le concept de "stratégie de comportement" qui a été utilisé avec profit dans le contexte du transport en commun. Nous supposons que les arcs du réseau possèdent des capacités rigides qui induisent, de façon endogène, des probabilités d'accès aux arcs. Les usagers du réseau basent leur comportement sur des stratégies qui consistent, en chaque sommet du résaeu en un ordre de préférence des arcs incidents à ce sommet. Bien que les stratégies soient des objets déterministes, leur réalisation au jour le jour, c'est-à-dire le trajet emprunté par chaque usager, est aléatoire. Ce modèle macroscopique complexe est analysé à la fois des points de vue théorique, algorithmique et numérique.

Mots-Clés: Équilibre. Affectation du trafic. Hyperchemin. Capacités. Inéquations variationnelles.

1 Introduction

Mathematical models of traffic assignment are usually based on Wardrop's principle [16] that states that, at equilibrium, flows are assigned to shortest paths with respect to current (flow dependent) travel delays. A variety of such models have been considered in the literature, including such features as several modes of transportation, variable demand, multiple user classes, stochasticity of travel delay perception and, more recently, the dynamic aspects of congestion. One feature of transportation networks that is frequently overlooked in most models is that links have finite capacities. Both in the static and dynamic cases, the capacities can be violated unless restrictive assumptions are made about the volume-delay relationships. For instance, in the static case, one might assume a functional form having a vertical asymptote at the capacity.

In the dynamic case, it has been shown by Astarita [1] that affine congestion functions are consistent with link capacities. Note that the latter conditions somewhat contradicts the former, which assumes an extremely nonlinear delay-flow relationship. Both posit functional relationships that might prove unrealistic. Another approach, first considered by Hearn [5] and next pursued by Larsson and Patriksson [8]), consists in considering explicitly within the model the capacity constraints. Although these constraints might prevent the existence of a Wardrop equilibrium, it can be proved that there exist Wardrop equilibria with respect to generalized delay functions that involve the dual variables associated with the capacity constraints. These dual variables have been interpreted as queueing delays occurring at the entrance of capacitated links (see Daganzo [3]).

A simple example will help to explain this approach. Consider a traffic assignment problem involving a single origin-destination couple with a demand of one unit of flow. Let us denote by P the set of origin-destination paths and by

$$x = \{x_p\}_{p \in P} \in X = \{x : \sum_{p \in P} x_p = 1, x \geq 0\}.$$

a feasible flow pattern. Each path p is endowed with a capacity u_p, which is equivalent to the minimum capacity of the arcs along path p. Let $F_p(x)$ be the travel time function along path p and let $F(x) = \{F_p(x)\}_{p \in P}$. If the capacities u_p are infinite, Wardrop's first principle states that an equilibrium is reached when travel costs are equal on all used paths, and this common cost λ is less than the actual cost on any unused path. Any equilibrium solution s can be characterized as a solution of the nonlinear complementarity problem:

$$\begin{aligned} x &\in X \\ F_p(x) - \lambda &\geq 0 \\ x_p[F_p(x) - \lambda] &= 0 \end{aligned}$$

or, equivalently, of the variational inequality:

$$\begin{aligned} x &\in X \\ \langle F(x), x - y \rangle &\geq 0 \qquad \forall y \in X. \end{aligned}$$

Now, in the capacitated case, it might be required to include a queueing delay α_p to a saturated arc in order to achieve the above equilibrium conditions. This yields the complementarity system:

$$x \in X \cap \{x \leq u\}$$

$$\begin{aligned} F_p(x) + \alpha_p - \lambda &\geq 0 \\ x_p[F_p(x) + \alpha_p - \lambda] &= 0 \\ \alpha_p)[x_p - u_p] &= 0 \\ \alpha_p &\geq 0, \end{aligned}$$

which is equivalent to the variational inequality:

$$\begin{aligned} & x \in X \cap \{x \leq u\} \\ & \langle F(x), x - y \rangle \geq 0 \qquad \forall y \in X \cap \{x \leq u\}. \end{aligned}$$

If the vector function F is the gradient of some function f, x is a solution of the previous variational inequality if and only if it is a first order stationary point for the mathematical program:

$$\min_{x \in X \cap \{x \leq u\}} f(x).$$

An advantage of the resulting mathematical program is that it can solved quite easily. However, the implicit use of the queueing delay resulting from transient overcapacity within a static assignment model is controversial. Indeed this implicit process is not mathematically described although its validity, using the concept of 'point queues', has been advocated by Daganzo [3].

Our approach to capacities is entirely different. We assume that the users of the transportation network select, at their origin node, a travel strategy. A strategy associates with each node of the network an ordered set of successor nodes. At each node, the user selects the first unsaturated arc within this set. Optimal strategies must clearly take into account the possibility that an arc be unavailable when the user reaches the tail node of that arc, and must propose an alternative arc. While strategies are deterministic, their realization depends on arc availability, and is therefore stochastic. Users state their preferences but cannot be assured, from day to day, of the actual path they will travel.

This approach has been previously applied with success in the realm of urban transit assignment (Nguyen and Pallottino [11], Spiess [14]). The mathematical concept underlying strategies is that of a hyperpath, or a path in a well-defined hypergraph. We believe that the field of application of this concept goes well beyond the realm of transit assignment, and allows a better representation of user behaviour in congested transportation networks. While our ultimate aim is to study priority networks, where the use of strategies as fundamental decision variables settles some paradoxical situations that arise when only path flow variables are considered, this work will focus on the implementation of the non-priority model which has been introduced by Marcotte and Nguyen [10].

The remainder of the paper is organized as follows. We start by giving a detailed description of a simple instance where no equilibrium expressed solely in terms of path flow variables exist, thus justifying our notion of equilibrium in capacitated networks. Then we formulate, in a rigorous manner, an equilibrium model with constant arc costs, i.e., where the sole congestion effects are induced by the link capacities. Next we present theoretical properties of this model. Finally we describe the implementation of a solution algorithm, and provide numerical results on test networks.

2 Strategic equilibria in capacitated networks

Defining equilibria meaningfully in capacitated transportation networks is a nontrivial task. To gain some insight into this matter, let us consider the graph illustrated in Figure 1, where each arc is endowed with a cost (shown next to the corresponding arc) and, possibly, a capacity (bracketed number). The paths from origin node 1 to destination node 5 are shown, together with their characteristics, in Table 1.

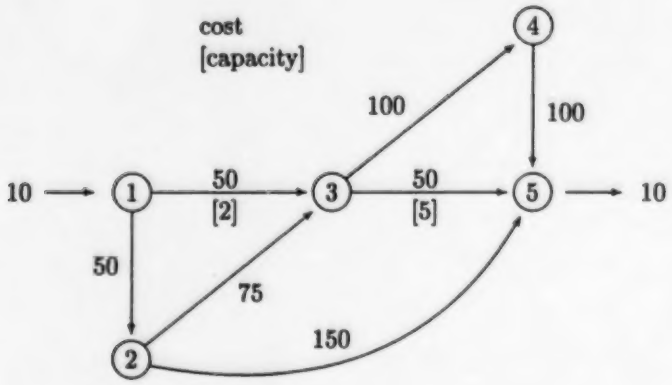

Figure 1: A capacitated network

path index	path	cost	capacity
1	$1-3-5$	100	2
2	$1-2-3-5$	175	5
3	$1-2-5$	200	∞
4	$1-3-4-5$	250	2
5	$1-2-3-4-5$	325	∞

Table 1: Network paths for a small example

Clearly, no Wardrop equilibrium compatible with this data exists, since the shortest path 1-3-5 cannot accommodate all the demand of 10 units. Quite naturally, one could settle for an extended Wardrop's principle expressed as:

> At equilibrium, there exists no path with positive residual capacity and a cost less than the cost of some path carrying positive flow.

In the above small example, any path flow vector of the form

$$x = (5-t, t, 5, 0, 0)$$

with $t \in [3,5]$ satisfies the above generalized Wardrop principle. However, whenever t is stricter greater than 3 and the users of the second path are endowed with the gift of prescience, it is tempting for them to switch to the less costly first path on which some capacity has been set free by themselves. In this sense, the equilibrium associated with the value $t = 3$ is more natural. Indeed this solution, associated with a queueing delay of 100 on arc (3,5), corresponds to a 'Hearn-Patriksson' equilibrium and is, notwithstanding the queueing delay, a system-optimal solution as well. The drawback of this approach is that

In the strategic approach, the behaviour of the network users is dictated by a rule that specifies, at each node of the transportation network, an ordered list of adjacent nodes. Such a set of strategies is given in Table 2. Some of these strategies, s_4 for instance, avoid capacitated arcs and correspond to ordinary paths. In contrast, a user adopting strategy s_1 could end up travelling on path 1, 2, 4 or 5, depending on the accessibility of the capacitated arcs (1,3) and (3,5).

node	1	2	3	4	5
s_1	[3,2]	[3]	[5,4]	[5]	[ ]
s_2	[3,2]	[5]	[5,4]	[5]	[ ]
s_3	[2]	[3]	[5,4]	[5]	[ ]
s_4	[2]	[5]	[ ]	[ ]	[ ]
s_5	[3,2]	[3]	[5]	[5]	[ ]
s_6	[3,2]	[5,3]	[5,4]	[5]	[ ]
s_7	[3]	[ ]	[5,4]	[5]	[ ]
s_8	[3,2]	[3]	[4,5]	[5]	[ ]

Table 2: A set of strategies for a small example

path	access probability	cost
$1-3-5$	$\frac{2}{10} \times \frac{5}{10} = \frac{1}{10}$	100
$1-3-4-5$	$\frac{2}{10} \times \frac{5}{10} = \frac{1}{10}$	250
$1-2-3-5$	$\frac{8}{10} \times \frac{5}{10} = \frac{4}{10}$	175
$1-2-3-4-5$	$\frac{8}{10} \times \frac{5}{10} = \frac{4}{10}$	325
$1-2-5$	0	200

Table 3: Path probabilities for a simple example

We make the fundamental assumption that, whenever the number x of users that want to access arc a exceeds the capacity u_a of a, the probability p of accessing the arc is set to $p = x/u_a$. This is equivalent to making the assumption that the users are independently and uniformly distributed at the tail node of arc a. These access probabilities allow us to compute the expected cost of the strategies. For instance, if all 10 users adopt strategy s_1, then the probability of accessing arc (1,2) is equal to 2/10, according to the formula $p = x/u_a$. The users, whether they access arc (1,2) or not, clash again at node 3, where the access probability of arc (3,5) is 5/10. These numbers, which are required to compute the access probabilities associated with the paths of the network, are shown in Table 3.

The expected value of each user's transportation delay is therefore

$$\left(\frac{1}{10} \times 100\right) + \left(\frac{1}{10} \times 250\right) + \left(\frac{4}{10} \times 175\right) + \left(\frac{4}{10} \times 325\right) + \left(\frac{0}{10} \times 325\right) = 235.$$

These probabilities could also have been computed with respect to the arcs (see [10] for a complete discussion and the interpretation of strategies as hyperpaths in a suitably defined hypergraph).

We argue that the assignment of all commuters to strategy s_1 is not rational, in the sense that strategy s_3, corresponding to the uncapacitated path $1-2-5$, is available at cost $200 < 235$. A **strategic equilibrium** is reached when all users are assigned to strategies of minimal expected delays. It is not too difficult to check on this small example that the unique equilibrium corresponds to the assigning 5 users to strategy s_1 and the remaining 5 to strategy s_2. The resulting strategic flows and path flows are shown in Table 4. One verifies that the expected delay of each strategy is equal to 185, which is less than the expected delay of unused strategies, thus fulfilling the equilibrium conditions.

In general, a vector x of **strategic flows** is a strategic equilibrium if and only if it is demand-feasible and satisfies the variational inequality

path	flow from s_1	flow from s_2	cost
$1-3-5$	$\frac{5}{6}$	$\frac{5}{6}$	100
$1-3-4-5$	$\frac{1}{6}$	$\frac{1}{6}$	250
$1-2-3-5$	$\frac{20}{6}$	0	175
$1-2-3-4-5$	$\frac{4}{6}$	0	325
$1-2-5$	0	4	200

Table 4: Equilibrium strategic path flows

$$\langle C(x), x-y\rangle \leq 0 \tag{1}$$

for every demand-feasible strategic vector y, where the component $C^s(x)$ of $C(x)$ is the expected delay associated with strategy s and the strategic vector x. Unfortunately, even on small examples as the one studied above, the cost function C is not available in closed form. Its evaluation requires requires the knowledge of total flows on the arcs of the network. These are readily available from path flows (via the arc-path incidence matrix) in standard traffic models. However, in the strategic model, the arc-hyperpath matrix, which is flow-dependant, is difficult to obtain. Therefore, in order to derive path flows from strategic flows, a loading algorithm, to be described in the next section, is required. Furthermore. this loading algorithm requires the knowledge of all strategies, which are in much larger number than the number of paths. However, as was shown by Marcotte and Nguyen [10], several strategies are dominated or equivalent and need not be considered. In the implementation section, we will see that the set of active strategies can be generated iteration by iteration.

The remaining sections of this paper will be devoted to the development of algorithms for computing strategic equilibria. The challenge posed by this goal is threefold:

- no natural assumption guarantees the monotonicity of the delay function C;
- the loading procedure which translates strategic flows into link flow is complex;
- 'best strategy' procedure must be substituted for a 'shortest path' procedure.

Implementation details will be provided, and numerical results on nontrivial networks will demonstrate the viability of this approach. We close this section with a list of the most frequently used notations. We slightly abuse language and write 'cost' and 'flow' in place of 'expected cost' and 'expected flow'.

$G = (N, A)$	network with node set N and arc set A
d_{qr}	demand from origin node q to destination node r
u_{jk}	capacity of arc (j, k)
$\bar{u}_{jk}$	residual capacity of arc (j, k)
S	set of strategies
$E_j(s)$	ordered list of nodes in the forward star j_+
$s = \{E_j(s)\}_j$	strategy
x_s	strategic flow
x	vector of strategic flows
z_j^s	flow from strategy having reached node j
$\pi_{jk}^s(x)$	probability of accessing node k from node j using strategy s
$C^s(x)$	cost of strategy s
$C(x)$	vector of strategic costs
W	set of active strategies (working set)

Other notation will be introduced as requested.

3 The loading mechanism

In traditional static traffic assignment models, arc flows can readily be derived form path flows via the information contained in the arc-path incidence matrix. Things are more complex in our model, since the arc-hyperpath incidence matrix is dependent on the strategic flow vector. In order to recover links flows, which are required to compute arc and path costs, one has to implement a nontrivial loading procedure, which is akin to a deterministic simulation. Throughout the paper, we make the assumption that this simulation is feasible, i.e., that there exists an assignment of arc flows that respects the capacities on the arcs of the network. This can be accomplished by introducing an artificial arc with high cost (a 'walk arc') for every origin-destination couple.

In the small examples considered so far, the preference sets of all users were identical at all tail nodes of capacitated arcs. This resulted in an intuitive and simple loading process. The generalization of this process to more complex situations, however, can lead to different arc flows, depending on the queue discipline adopted, i.e., the way one manages the simultaneous assignment of flows to the capacitated arcs of the network. We consider two such disciplines[1] :

single file processing The users are randomly and uniformly distributed in a single vertical file and no overpassing is allowed;

parallel processing The users are randomly and uniformly distributed in a single vertical file. A user is allowed to access an arc if its residual capacity is positive and if no user with the same current choice stands in front of her in the queue. In other words, a user may overpass another user who is still waiting for a higher priority arc (with positive residual capacity) within her own strategy. However, the overpassed user will retain her 'priority number' when competing for an outgoing arc. This rule implicitly assumes that the users' behaviour is dictated by the 'status' of a capacitated arc (saturated or unsaturated) and not by its residual capacity. Equivalently, one could say that they are unaware of the number of competing users lying ahead in the queue.

[1] While the analogy is made with queues involving discrete customers, the loading procedure acts on continuous of vehicles.

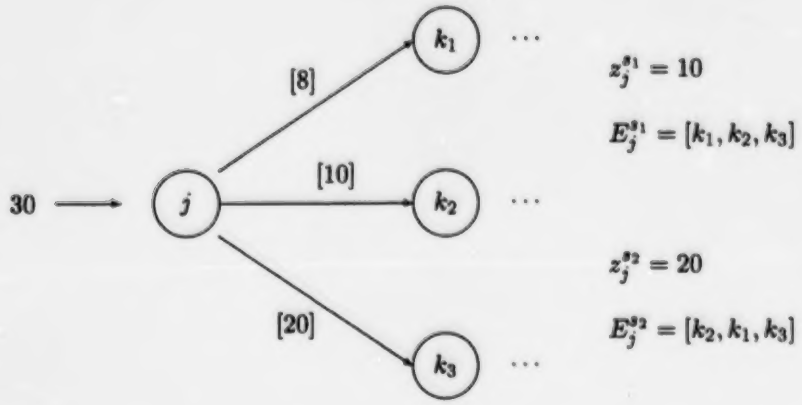

Figure 2: Loading process at node j

We believe that the single file discipline is better suited to the situation of transportation networks. Fortunately it is computationally more tractable and also possesses interesting theoretical properties which are lacking in the parallel case, as we will see later.

The differences between the two disciplines will be illustrated on the example of Figure 2

Let us first consider the single file processing. At the first iteration of the loading process, the strategic flows x^{s_1} and x^{s_2} are assigned to their respective preferred node at a rate proportional to their respective size. Since the ratio 10/20 is less than the ratio 8/10, the arc (j, k_2) gets saturated after half the flow of strategy s_2 is assigned to it. At that instant, half the strategic flow from s_1 (5 units) has reached node k_1. Next, node k_2 is removed from the preference sets, the arc capacities are replaced by their residual capacities and the process is repeated. At the second iteration, 5 units of flow from s_1 and 10 from s_2 compete for the residual capacity (3) of arc (j, k_1). According to our proportionality assumption, 1 flow unit from s_1 reaches node k_1 while 2 flow units from s_2 reach k_2. A third iteration is required to load the unassigned flow to the uncapacitated arc (j, k_3). The arc access probabilities are then obtained by computing the ratio of the of 'successful flow' over the corresponding arc capacity. These calculations are summarized in Table 5.

iteration	arc	(j, k_1)	(j, k_2)	(j, k_3)
1	residual capacity	8	10	20
	flow	$5(s_1)$	$10(s_2)$	0
2	residual capacity	3	0	20
	flow	$5(s_1)+1(s_1)+2(s_2)$	$10(s_2)$	0
3	residual capacity	0	0	20
	flow	$6(s_1)+2(s_2)$	$10(s_2)$	$4(s_1)+8(s_2)$

probabilities $\pi_{jk_1}^{s_1} = \frac{6}{10}$ $\pi_{jk_3}^{s_1} = \frac{4}{10}$ $\pi_{jk_1}^{s_2} = \frac{2}{20}$ $\pi_{jk_2}^{s_2} = \frac{10}{20}$ $\pi_{jk_3}^{s_2} = \frac{8}{20}$

Table 5: Single file loading at node j

At the first iteration of the parallel loading procedure, the flow is assigned in parallel to the outgoing

iteration	arc	(j,k_1)	(j,k_2)	(j,k_3)
1	residual capacity	8	10	20
	flow	$8(s_1)$	$8(s_2)$	0
2	residual capacity	0	2	20
	flow	$8(s_1)$	$8(s_2)+\frac{2}{7}(s_1)+\frac{12}{7}(s_2)$	0
3	residual capacity	0	0	20
	flow	$8(s_1)$	$\frac{68}{7}(s_2)+\frac{2}{7}(s_1)$	$\frac{12}{7}(s_1)+\frac{72}{7}(s_2)$

probabilities: $\pi^{s_1}_{jk_1}=\frac{8}{10}$ $\pi^{s_1}_{jk_2}=\frac{2}{7\times 10}$ $\pi^{s_1}_{jk_3}=\frac{12}{7\times 10}$ $\pi^{s_2}_{jk_2}=\frac{68}{7\times 20}$ $\pi^{s_2}_{jk_3}=\frac{72}{7\times 20}$

Table 6: Parallel loading at node j

nodes k_1 and k_2, at identical rates. Arc (j,k_1) gets saturated first. Next, node k_1 is removed from the preference sets $E^{s_1}_j$ and $E^{s_2}_j$ and the residual capacities are updated. At the second iteration, 14 units of flow are assigned to the remaining capacity (2) of arc (j,k_2) in respective proportions 2 (strategy s_1) and 12 (strategy s_2). Details of the calculations are shown in Table 6.

In order that a feasible flow assignment exist, we require that the network be acyclic. The loading is performed in accordance with a topological ordering of the nodes. While the concepts underlying the algorithm are intuitive, its precise description is quite involved. We give below the pseudocode performing the assignment of a strategic flow vector $x=\{x_s\}_{s\in W\subset S}$ under the single file processing rule.

The working set W might include strategic flows with zero values. It is actually important to consider zero-flow strategies since the best strategy, for fixed strategic flow vector x, will most likely be such a strategy. Since the access probabilities of a zero-flow strategy cannot be computed using its flow values, an alternative approach has to be adopted. Let us introduce ρ^s_j, the proportion of strategic flow from s that could reach node j. At the origin node of strategy s, this proportion is set to 1. The arc access probabilities π^s_{jk} are then iteratively computed according to the formula

$$\pi^s_{jk} \leftarrow \pi^s_{jk} + \beta^s \rho^s_j$$

where β^s is a local variable equal to access probability to node k. Next, the residual proportion is updated according to the formula $\rho^s_j \leftarrow \rho^s_j(1-\beta^s)$ and the process is halted whenever ρ^s_j becomes equal to zero. This assessment of zero flow strategy will be most important in the algorithm that produces a best strategy, to be described in the next section.

PROCEDURE CAPLOAD (x)

input: $x=\{x_s\}_{s\in W}$ *[strategic flow vector]*
output: $\pi=\{\pi^s_{jk}\}_{(j,k)\in A, s\in W}$ *[conditional arc access probabilities]*
$C=\{C^s\}_{s\in W}$ *[strategic cost vectors]*

INITIALIZATION
for $s\in W$ **do**
$C^s \leftarrow 0$ *[cost of strategy s]*
$z^s_{q(s)} \leftarrow x_s$ *[flow at origin node of s]*
$\tau^s_{q(s)} \leftarrow 1$ *[probability of accessing origin node of j]*
for $j\in N$ $(j\neq q(s))$ **do**

```
        z_j^s ← 0                                   [strategic flow at node j]
        τ_j^s ← 0                                   [probability of accessing node j
        for k ∈ E_js do                                 using strategy s]
            v̄_jk^s ← 0                              [strategic flow on arc (j,k)]
            π_jk^s ← 0                              [conditional probability
        endfor                                          of accessing arc (j,k) using strategy s]
    endfor
endfor

ASSIGNMENT PHASE
for j ∈ N (according to topological order) do
    W̄ ← {s ∈ W : E_js ≠ ∅}                          [set of active strategies]
    for s ∈ W̄ do
        Ē_j^s ← E_js                                [updated preference order]
        z̄_j^s ← z_j^s                               [residual strategic flow]
        ρ_j^s ← 1                                   [residual probability]
    endfor
    for k ∈ j^+ do
        ū_jk ← u_jk                                 [residual capacity]
    endfor
    while W̄ ≠ ∅ do
        K ← ∅                                       [set of residual first choices]
        for k ∈ j^+ do
            d_k ← 0                                 [demand for node k]
            S_k ← ∅                                 [set of strategies having
        endfor                                          node k as first residual choice]
        for s ∈ W̄ do
            k ← Ē_j^s(1)                            [first residual choice of strategy s]
            K ← K ∪ {k}                             [construction of set K]
            S_k ← S_k ∪ {s}                         [construction of S_k]
            d_k ← d_k + z̄_j^s                       [construction of d_k]
        endfor
        ν ← max{d_k/ū_jk : k ∈ K}
        β ← min{1/ν, 1}                             [proportion of affected residual flow]
        for k ∈ K do
            for s ∈ S_k do
                z_k^s ← z_k^s + βz̄_j^s              [updating flow z_k^s]
                v̄_jk^s ← v̄_jk^s + βz̄_j^s            [updating arc flow v̄_jk^s]
                z̄_j^s ← z̄_j^s − βz̄_j^s              [updating residual flow z̄_j^s]
                π_jk^s ← π_jk^s + βρ_j^s            [updating probability π_jk^s]
                ρ_j^s ← (1 − β)ρ_j^s                [updating proportion ρ_j^s]
            endfor
            ū_jk ← ū_jk − βd_k                      [updating residual capacity ū_jk]
            if ū_jk = 0 then
              for s ∈ W̄ do
                  Ē_j^s ← Ē_j^s − {k}               [updating ordering of Ē_j^s]
              endfor
```

```
            endif
          endfor
          if β = 1 then
            W̄ = ∅                                   [end of loading]
          endif
      endwhile
      for s ∈ W do
          for k ∈ E_js do
              C^s ← C^s + c_jk τ_j^s π_jk^s          [update of C^s]
              τ_k^s ← τ_k^s + τ_j^s π_jk^s           [update of probability τ_k^s]
          endfor
      endfor
  endfor
```

At each step within the **while** structure an arc becomes saturated. Therefore this loop is executed at most $|j^+|$ times for each node j, where j^+ denotes the forward star of node j. It follows that the loop is executed at most $\sum_{j \in N} |j^+| = |A|$ times and that the total running time of the algorithm is polynomial.

The overall structure of the algorithm corresponding to the parallel case is similar. The main difference lies in the definition of the probability β and in the treatment of the null demand case. The modifications occur within the **while** loop, whose pseudocode is given below.

```
while W̄ ≠ ∅ do
      K ← ∅                                        [set of residual first choices]
      for k ∈ j^+ do
          d_k ← 0                                  [demand for node k]
          S_k ← ∅                                  [set of strategies having
                                                       node k as first residual choice]
      endfor
      for s ∈ W̄ do
          k ← Ē_j^s(1)                             [first residual choice of strategy s]
          K ← K ∪ {k}                              [construction of the set K]
          S_k ← S_k ∪ {s}                          [construction of the set S_k]
          d_k ← d_k + z̄_j^s                        [construction of d_k]
      endfor
      d̄_min ← min{d_k : k ∈ K}                     [minimum demand]
      ū_min ← min{ū_jk : k ∈ K}                    [minimum residual capacity]
      μ ← min{d̄_min, ū_min}                        [outgoing flow]
      if μ ≠ 0 then
        for k ∈ K do
            β ← μ/d_k                              [proportion of residual flow assigned
            for s ∈ S_k do                             at node k]
                z_k^s ← z_k^s + β z̄_j^s            [update of flow z_k^s]
                v̄_jk^s ← v̄_jk^s + β z̄_j^s          [update of arc flow v̄_jk^s]
                z̄_j^s ← z̄_j^s − β z̄_j^s            [update of residual flow z̄_j^s]
                π_jk^s ← π_jk^s + β ρ_j^s          [update of access probability π_jk^s]
                ρ_j^s ← (1 − β) ρ_j^s              [update of proportion ρ_j^s]
                if ρ_j^s = 0 then
                  W̄ ← W̄ − {s}                      [deletion of strategy s]
                endif
```

```
            endfor
            ū_jk ← ū_jk − βd_k                [update of residual capacity ū_jk]
            if ū_jk = 0 then
              for s ∈ W̄ do
                  Ē_j^s ← Ē_j^s − {k}         [update of the order set Ē_j^s]
              endfor
            endif
        endfor
            else                              [degenerate case μ = 0]
        for k ∈ K do
            if d_k = 0 then
              for s ∈ S_k do
                  π_jk^s ← π_jk^s + ρ_j^s     [update of the probability π_jk^s]
                  W̄ ← W̄ − {s}                 [deletion of strategy s]
              endfor
            endif
        endfor
      endif
endwhile
```

In the parallel case, it is no longer true that an arc gets saturated at each iteration of the **while** loop. An upper bound on the number of iterations is given by

$$\sum_{j \in N} (|j^+| + |W|) = |A| + |W||N|,$$

which is dependent on the size $|W|$ of the working set. This bound is no longer polynomial in terms of the number of arcs $|A|$ of the network G.

We close this section with a result stating the equivalence of the two disciplines at a node with at most two outgoing arcs.

Proposition 1 *The arc access probabilities $\{\pi_{jk}^s\}_{k,s}$ derived from the loading process are identical under both queue disciplines provided that the number of successors of node j is at most 2.*

Proof Let k_1 and k_2 be the two successors of node j and let

u_1 : capacity of arc (j, k_1)
u_2 : capacity of arc (j, k_2)
z_1 : strategic flow at node j for strategy s_1
z_2 : strategic flow at node j for strategy s_2.

To avoid the trivial case where both strategies are identical, we assume that $E_j^{s_1} = [k_1, k_2]$ and $E_j^{s_2} = [k_2, k_1]$. In order that the loading be feasible, we further assume that $z_1 + z_2 \leq u_1 + u_2$. We consider three mutually exclusive cases:

1. $z_1 \leq u_1$ and $z_2 \leq u_2$

   Trivially, $\pi_{jk_1}^{s_1} = \pi_{jk_2}^{s_2} = 1$ in both cases.

node	1	2	3	4	5
flow	40	20	25	15	40

Table 7: Node flows corresponding to the loading of strategy s_1

arc	(1,2)	(1,3)	(1,4)	(2,3)	(2,5)	(3,4)	(3,5)	(4,5)
flow	20	10	10	15	5	20	15	

Table 8: Arc flows corresponding to the loading of strategy s_1

2. $z_1 > u_1$ and $z_2 \leq u_2$

   Since arc (j, k_1) gets saturated first, u_1 units of flow from strategy s_1 are assigned to arc (j, k_1). The remainder of the flow is assigned to arc (j, k_2). For both queue disciplines, we obtain $\pi^{s_1}_{jk_1} = u_1/*z_1$, $\pi^{s_1}_{jk_2} = (z_1 - u_1)/z_1$ and $\pi^{s_2}_{jk_2} = 1$.

3. $z_1 \leq u_1$ and $z_2 > u_2$

   This is the symmetric case of case 2.

4 Computing a best strategy

All algorithms that solve standard traffic assignment problem rely on a shortest path procedure. When dealing with strategic flows, things become significantly more complex, since the cost function is not available in closed form. Of course, computing the expected cost of an existing strategy is straightforward. The challenge lies in pricing out of strategies outside the working set W. For a given strategic flow vector x inducing arc access probabilities $\pi^s_{jk}(x)$, an optimal strategy $\overline{s}$, characterized by its preference sets $\{E^{\overline{s}}_j\}_{j \in N}$, is obtained from the backward recursion

$$C^{\overline{s}}_j = \begin{cases} \infty & \text{if } j > r \\ 0 & \text{if } j = r \\ \sum_{k \in E^{\overline{s}}_j} \pi^{\overline{s}}_{jk}(c_{jk} + C^{\overline{s}}_k) & \text{if } j < r, \end{cases} \tag{2}$$

where C^s_j denotes the expected travel cost from node j to the destination node r and where $E^{\overline{s}}_j$, the optimal preference order at node j, sorted in increasing order of the neighbouring nodes' labels $C_{jk} + C^{\overline{s}}_k$. Note that a similar recursion has been used by Nguyen and Pallottino [12] to compute shortest hyperpaths in a transit assignment model. The main difference with their algorithm is that, in the transit case, the access probabilities are not flow dependent.

Before stating the algorithm, we will illustrate the computations on the simple example of Figure 3 where a demand of 40 units has to be assigned from node 1 to node 5. Next to each arc is indicated its cost and, within square brackets, its capacity. Let us start with a working set containing the sole strategy

$$s_1 = ([2,4,3],[5,3],[5,4],[5],[\ ]).$$

Under both file disciplines, 20 units reach node 2 from node 1. From these 20 units, 10 units reach node 4 and the remaining 10 reach node 3. At node 2, the 20 units are assigned to arcs (2,5) and (2,3) in the respective proportions 1/4 and 3/4. Thus 25 (10+15) units of flow access node 3. Only a fifth of these units cannot access the arc (3,5) directly and must transit through node 4. The flows reaching the nodes of the network are given in Table 7; the arc flows are given in Table 8. The expected cost is equal to 47.5.

Now let us compute an optimal strategic answer $\overline{s}$ (not to be confused with an equilibrium solution) in the single file case. Starting from the destination node, we set $C^{\overline{s}}_5 = 0$ and $E^{\overline{s}}_5 = [\ ]$. At node 4, the

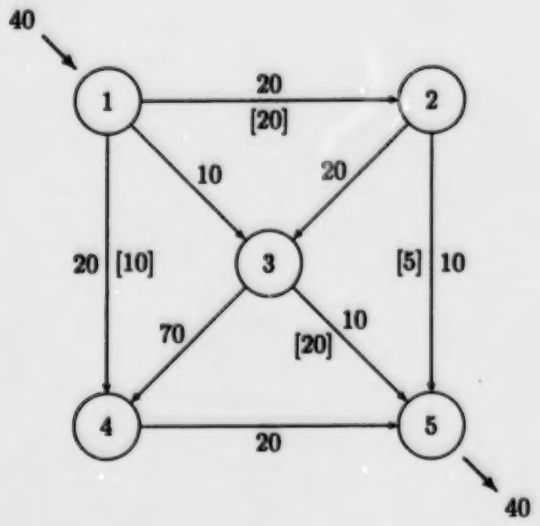

Figure 3: Computing a best strategy: example network

optimal preference order is trivially $E_4^{\bar{s}} = [5]$ and $C_4^{\bar{s}} = 20$. At node 3, the choice of $(3,5)$ as preferred arc is optimal and $E_3^{\bar{s}} = [5,4]$. The loading of strategy s^* at node 3 yields the probabilities $\pi_{35}^{\bar{s}} = 4/5$ and $\pi_{34}^{\bar{s}} = 1/5$. By applying the backward recursion at node 3, we obtain

$$C_3^{\bar{s}} = \frac{4}{5}(10 + C_5^{\bar{s}}) + \frac{1}{5}(70 + C_4^{\bar{s}}) = 26.$$

The optimal preference order at node 2 is clearly $E_2^{\bar{s}} = [5,3]$. The access probabilities, already computed for strategy s_1, are $\pi_{25}^{\bar{s}} = 1/4$ and $\pi_{23}^{\bar{s}} = 3/4$ respectively. We obtain

$$C_2^{\bar{s}} = \frac{1}{4}(10 + C_5^{\bar{s}}) + \frac{3}{4}(20 + C_3^{\bar{s}}) = 37.$$

The process terminates at the origin node where the preference order is dependent on the value of the strategic flow s_1. The optimal ordering $E_1^{\bar{s}} = [3]$ minimizes the expected cost

$$C_1^{s} = \pi_{12}^{s}(20 + C_2^{\bar{s}}) + \pi_{13}^{s}(10 + C_3^{\bar{s}}) + \pi_{14}^{s}(20 + C_4^{\bar{s}}) = 57\pi_{12}^{s} + 36\pi_{13}^{s} + 40\pi_{14}^{s}.$$

Indeed, since arc $(1,3)$ has infinite capacity, we have $\pi_{13}^{\bar{s}} = 1$. The corresponding optimal strategy is $\bar{s} = ([3],[5,3],[5,4],[5],[\ ])$, with expected cost 36. The results of this procedure is summarized in Table 9.

In this example, the best strategic answer turns out to be $\bar{s}$ in the parallel case as well. At each node, the optimal preference order is obtained by sorting, in decreasing order, the quantities $c_{jk} + C_k^{\bar{s}}$ for all nodes in j^+. As we will prove in subsection this local strategy is always optimal in the single file case. However this does need be the case if the parallel queue discipline is assumed. Indeed consider the loading that takes place at node j in the example of Figure 4. By sorting the nodes k_1, k_2 and k_3 according to the preceding criterion we expect the preference order to be $[k_1, k_2, k_3]$. The cost of this strategy $\bar{s}$ resulting from the zero flow loading procedure is

$$C_j^{\bar{s}} = \left(\frac{10}{25} \times 25\right) + \left(\frac{5}{25} \times 30\right) + \left(\frac{10}{25} \times 45\right) = 35.$$

node j	$E_j^{\bar{s}}$	$C_j^{\bar{s}}$
5	[ ]	0
4	[5]	20
3	[5,4]	26
2	[5,3]	37
1	[3]	36

Table 9: The optimal strategy for the small example

Now consider the preference order $E_j^{s_2} = [k_2, k_3]$. The loading of s_2 at node j yields $\pi_{jk_2}^{s_2} = 1$ and $C_j^{s_2} = 30$, which is better than $\bar{s}$. This shows that the determination of the optimal strategy at a given node is a task that is not solvable by a greedy technique but might require the enumeration of all possible permutations of the outgoing arcs at node j.

The procedure CAPSHORT, which returns, for a given strategic flow and origin-destination pair (q, r), an optimal strategy together with its expected cost, is listed below.

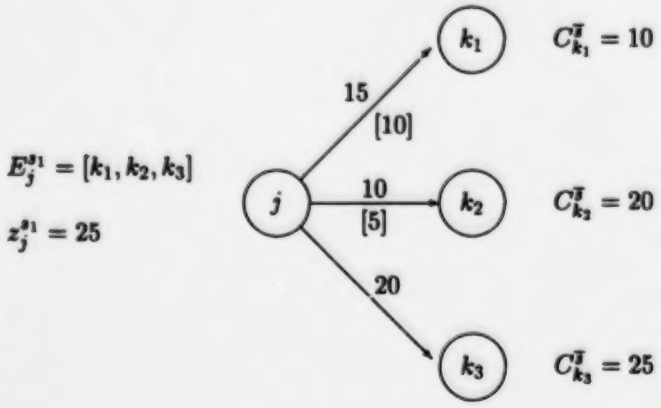

Figure 4: Constructing the set $E_j^{\bar{s}}$ (parallel processing)

PROCEDURE CAPSHORT $(z, (q,r))$

input: $z = \{z_j^s\}_{s \in W, j \in N}$ *[strategic flow vector at the nodes]*
$(q,r) \in L$ *[O-D pair]*
output: $\bar{s}$, $C^{\bar{s}}$ *[optimal strategy and its cost]*

INITIALIZATION
for $j \in N$ **do**
  if $j \leq r$ **then** $C_j^{\bar{s}} \leftarrow 0$ *[cost of optimal strategy*
    else $C_j^{\bar{s}} \leftarrow \infty$ *leaving node j]*
  endif
endfor

MICRO-LOADING (loading of a zero flow strategy)
for $j \in N$ $j < r$ (in reverse topological order) **do**
  $\overline{E}_j^s \leftarrow \text{sort}(j^+)$ *[sort in increasing order the set $\{c_{jk} + C_k^{\bar{s}}\}_{k \in j^+}$]*
  for $k \in \overline{E}_j^s$ **do**
    $\overline{u}_{jk} \leftarrow u_{jk}$ *[residual capacity of arc (j,k)]*
    $\pi_{jk}^{\bar{s}} \leftarrow 0$ *[conditional access probability to*
  endfor *arc (j,k) under strategy $\bar{s}$]*
  $\overline{W} \leftarrow \{s \in W : E_{js} \neq \emptyset\} \cup \{\bar{s}\}$ *[working set]*
  for $s \in \overline{W}$ **do**
    $\overline{E}_j^s \leftarrow E_{js}$ *[residual preference order]*
    $\overline{z}_j^s \leftarrow z_j^s$ *[residual strategic flow]*
    $\rho_j^s \leftarrow 1$ *[proportion of residual strategic flow]*
  endfor
  while $\rho_j^{\bar{s}} \neq 0$ **do**
    $K \leftarrow \emptyset$ *[set of residual first choices]*
    for $k \in j^+$ **do**
      $d_k \leftarrow 0$ *[demand for node k]*
      $S_k \leftarrow \emptyset$ *[set of strategies having*
    endfor *node k as first residual choice]*
    for $s \in \overline{W}$ **do**
      $k \leftarrow \overline{E}_j^s(1)$ *[first residual choice of strategy s]*
      $K \leftarrow K \cup \{k\}$ *[construction of K]*
      $S_k \leftarrow S_k \cup \{s\}$ *[construction of S_k]*
      $d_k \leftarrow d_k + \overline{z}_j^s$ *[construction of d_k]*
    endfor
    $\nu \leftarrow \max\{d_k / \overline{u}_{jk} : k \in K\}$
    $\beta \leftarrow \min\{1/\nu, 1\}$ *[proportion of assigned residual flow]*
    for $k \in K$ **do**
      for $s \in S_k$ **do**
        if $s = \bar{s}$ **then**
          $\pi_{jk}^s \leftarrow \pi_{jk}^s + \beta \rho_j^s$ *[updating access probability $\pi_{jk}^{\bar{s}}$]*
          $\rho_j^s \leftarrow (1 - \beta)\rho_j^s$ *[updating proportion $\rho_j^{\bar{s}}$]*
            else
          $\overline{z}_j^s \leftarrow \overline{z}_j^s - \beta \overline{z}_j^s$ *[update of the residual flow $\overline{z}_j^s$]*

```
                    ρ_j^s ← (1 − β)ρ_j^s        [updating the proportion ρ_j^s]
                  endif
                endfor
                ū_jk ← ū_jk − βd_k              [update of the residual capacity ū_jk]
                if ū_jk = 0 then
                  for s ∈ W̄ do
                    Ē_j^s ← Ē_j^s − {k}         [updating the residual order Ē_j^s]
                  endfor
                endif
              endfor
              if β = 1 then                     [end of micro loading]
                ρ_j^s̄ = 0
              endif
        endwhile
        for k ∈ E_j^s̄ do
            C_j^s̄ ← C_j^s̄ + π_jk^s̄(c_jk + C_k^s̄)   [update of cost C_j^s̄]
        endfor
        if j = q then C^s̄ ← C_q^s̄               [cost of optimal strategy]
        endif
endfor
```

In the parallel case, the **while** loop is modified to take into account the different loading rule. The description below sticks to the greedy method for computing a good strategy. In order to guarantee the optimality of the computed strategy, one should consider all possible preference sets at each node of the network.

```
while W̄ ≠ ∅ do
      K ← ∅                                 [set of residual first choices]
      for k ∈ j^+ do
          d_k ← 0                           [demand for node k]
          S_k ← ∅                           [set of strategies having
                                                node k as first residual choice]
      endfor
      for s ∈ W̄ do
          k ← Ē_j^s(1)                      [first residual choice of strategy s]
          K ← K ∪ {k}                       [construction of K]
          S_k ← S_k ∪ {s}                   [construction of S_k]
          d_k ← d_k + z̄_j^s                 [construction of d_k]
      endfor
      d̄_min ← min{d_k : k ∈ K}              [minimum demand]
      ū_min ← min{ū_jk : k ∈ K}             [minimum residual capacity]
      μ ← min{d̄_min, ū_min}                 [assigned flow]
      if μ ≠ 0 then
        for k ∈ K do
            β ← μ/d_k                       [proportion of assigned strategic flow
            for s ∈ S_k do                      at node k]
                if s = s̄ then
                  π_jk^s ← π_jk^s + βρ_j^s  [update of probability π_jk^s̄]
```

```
                    ρ_j^s ← (1 − β)ρ_j^s          [update of proportion ρ_j^s]
                        else
                    x̄_j^s ← x̄_j^s − βx̄_j^s        [update of residual flow x̄_j^s]
                    ρ_j^s ← (1 − β)ρ_j^s          [update of proportion ρ_j^s]
                  endif
                  if ρ_j^s = 0 then
                    W̄ ← W̄ − {s}                   [deletion of strategy s]
                  endif
              endfor
              ū_jk ← ū_jk − μ                     [update of residual capacity ū_jk]
              if ū_jk = 0 then
                for s ∈ W̄ do
                    Ē_j^s ← Ē_j^s − {k}           [update of residual order Ē_j^s]
                endfor
              endif
          endfor
              else                                [degenerate case: μ = 0]
          for k ∈ K do
              if d_k = 0 then
                for s ∈ S_k do
                    π_jk^s ← π_jk^s + ρ_j^s       [update of probability π_jk^s]
                    ρ_j^s ← 0                     [update of proportion ρ_j^s]
                    W̄ ← W̄ − {s}                   [deletion of strategy s]
                endfor
              endif
          endfor
      endif
endwhile
```

5 The equilibrium model

By definition, a strategic vector x^* is in equilibrium if it lies in the set X of demand-feasible vectors and satisfies the variational inequality $VI(C, X)$

$$\langle C(x^*), x^* - x \rangle \leq 0 \qquad \forall y \in X.$$

Since the cost mapping C is not available in closed form, the study of the theoretical properties of $IV(C, X)$ is not an easy task. Even though all the model is driven by simple parameters, namely travel costs (or arc lengths) and capacities, the resulting model is nonlinear and asymmetric! In this section we will exhibit a list of properties that the model possess or do not possess.

5.1 A 'Braess paradox'

In a standard model of traffic assignment with nonconstant arc costs, it might occur that an improvement to the network, e.g. the addition of an arc, results in an increase of the travel times of *all* users of the network. This situation is known in the literature as the 'Braess paradox', in the reference to the simple example of such a situation presented by Braess [2].

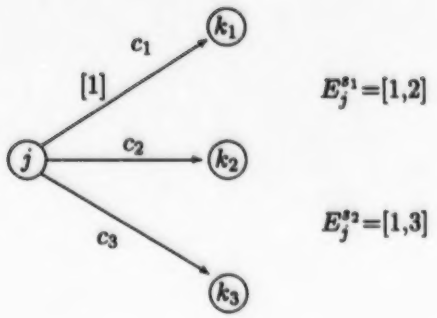

Figure 5: An example of nonintegrability

In the strategic model, this situation may also arise. Indeed, reconsider the capacitated network of Figure 1, where the expected equilibrium cost was 185. If the arc (2,3) is deleted from the network, the entire equilibrium flow is redirected to the strategy $s = ([3,2],[5],[5,4],[5],[\ ])$ whose expected cost $\frac{2}{10}100 + \frac{8}{10}200 = 180$ is less than 185. This shows that a decrease in an arc capacity may lead to an improvement in the expected travel times of *all* users. This should not be surprising: equilibrium and optimality are two quite different concepts.

5.2 Integrability

In general, the mapping C is not a gradient mapping. This can be observed on the small subnetwork of Figure 5. If the sum $x_1 + x_2$ of the strategic flows is greater than the capacity of the first arc, one has, in both the single file and parallel cases:

$$\begin{aligned} C_1(x_1,x_2) &= \frac{c_1}{x_1+x_2} + \left(1 - \frac{1}{x_1+x_2}\right) c_2 \\ C_2(x_1,x_2) &= \frac{c_1}{x_1+x_2} + \left(1 - \frac{1}{x_1+x_2}\right) c_3. \end{aligned}$$

The cross partial derivatives are given by the expressions

$$\begin{aligned} \frac{\partial C_1}{\partial x_2}(x_1,x_2) &= \frac{c_2 - c_1}{(x_1+x_2)^2} \\ \frac{\partial C_2}{\partial x_1}(x_1,x_2) &= \frac{c_3 - c_1}{(x_1+x_2)^2} \end{aligned}$$

and are clearly distinct if $c_3 \neq c_1$. Hence, the costs mapping C lacks the symmetry assumption which is required to make it a gradient mapping, and the variational inequality $VI(C,X)$ cannot be reformulated as a 'standard' optimization problem over the feasible set X.

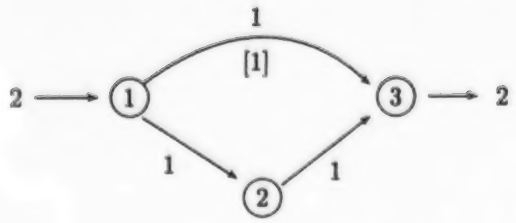

Figure 6: An example of nondifferentiability

5.3 Differentiability

Let us consider the situation illustrated in Figure 6. The two strategies of interest are $s_1 = ([3,2],[3],[\ ])$ and $s_2 = ([2],[3],[\ ])$. It is not difficult to check that the cost function is

$$C(x_1, x_2) = (\max\{1, 2 - \frac{1}{x_1}\}, 2),$$

whose first partial derivative with respect to x_1 does not exist when $x_1 = 1$ (the left derivative is zero while the right derivative is equal to 3). Note that in this example, the single file and parallel processings yield identical flows and access probabilities.

5.4 Continuity

The mapping C is continuously dependent on the access probabilities π_{ij}^{kl} which are defined as ratios of residual capacities over flows. Whenever the degenerate situation where arcs get saturated simultaneously does not occur, these probabilities are continuous functions of the strategic flow vectors. If such a degenerate case arises, discontinuities in the cost function may arise in the parallel case[2].

The following example (see Figure 7) show that, in the parallel case, the cost function may be discontinuous in the degenerate case. The only interesting three strategies are the following:

node	1	2	3	4
s_1	[2,4,3]	[4,3]	[4]	[ ]
s_2	[4,3]	[ ]	[4]	[ ]
s_3	[4,2,3]	[4,3]	[4]	[ ]

Assume that $x_1 \leq 8$. At node 1, x_1 flow units prefer node 2, while $x_2 + x_3$ select 4 as their preferred outgoing node. The access probabilities are

$$\pi_{12}^{s_1} = 1,$$

$$\pi_{14}^{s_2} = \pi_{14}^{s_3} = \frac{12}{x_2 + x_3}.$$

Next, the proportion of x_3 that accesses arc $(1,2)$ is

$$\beta = \begin{cases} 0 & \text{if } x_1 = 8 \text{ and } x_3 = 0 \\ \min\{1, (8 - x_1)/(x_3(1 - \pi_{14}^{s_3}))\} & \text{otherwise.} \end{cases}$$

[2]A counterexample in the single file case could not be found. We conjecture that C is continuous in that case.

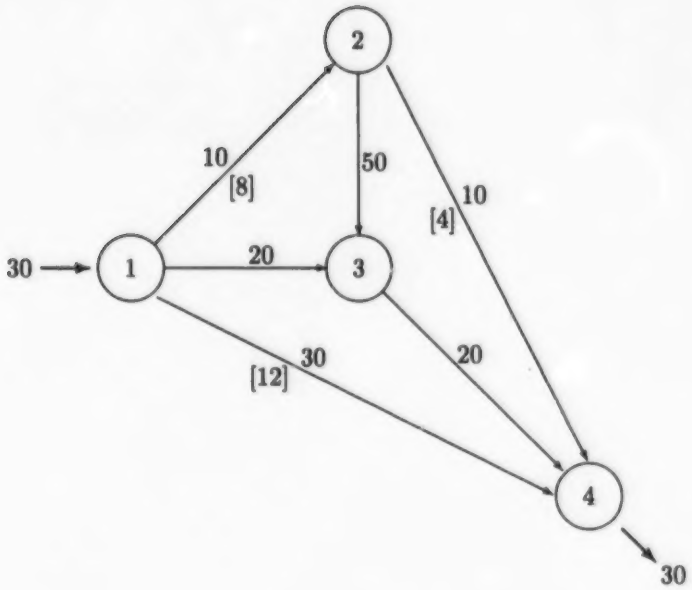

Figure 7: An example of discontinuity

Therefore, the probability of accessing $(1,2)$ under strategy s_3 is

$$\pi_{12}^{s_3} = \beta\,(1 - \pi_{14}^{s_3})\,.$$

The assignment of the residual flow associated with strategies s_2 and s_3 at node 3 ends the loading process at the first node. The total flow that has reached node 2 at this stage is

$$\lambda = x_1 + \pi_{12}^{s_3}\, x_3\,.$$

The access probability to node $(2,4)$ under strategies s_1 and s_3 is now

$$\pi_{24}^{s_1} = \pi_{24}^{s_3} = \min\{1, 4/\lambda\}.$$

The end of the loading process is straightforward and, based on the access probabilities, we obtain the cost function

$$\begin{aligned}
C_1 &= 10 + (\pi_{24}^{s_1} \times 10) + (1 - \pi_{24}^{s_1}) \times 70 \\
&= 80 - 60\,\pi_{24}^{s_1}\,, \\
C_2 &= (\pi_{14}^{s_2} \times 30) + (1 - \pi_{14}^{s_2}) \times 40 \\
&= 40 - 10\,\pi_{14}^{s_2}\,, \\
C_3 &= (\pi_{14}^{s_3} \times 30) + \pi_{12}^{s_3}\,(10 + (\pi_{24}^{s_3} \times 20) + (1 - \pi_{24}^{s_3}) \times 70) + (1 - \pi_{14}^{s_3} - \pi_{12}^{s_3}) \times 40 \\
&= 40 - 10\,\pi_{14}^{s_3} + 40\,\pi_{12}^{s_3} - 60\,\pi_{12}^{s_3}\,\pi_{24}^{s_3}\,.
\end{aligned}$$

We will now show that the cost function is not continuous at the point $(8, 22, 0)$. Consider the strategic flow vector $(8 - \epsilon, 22, \epsilon)$ for some small positive number ϵ. Under strategy s_3, the probability of accessing

arc $(1,4)$ is:

$$\pi_{14}^{s_3} = \frac{12}{22+\epsilon}.$$

After arc $(1,4)$ gets saturated, the residual flow of s_3 (which is less than ϵ) is assigned to arc $(1,2)$ whose residual capacity is exactly ϵ. We obtain

$$\pi_{12}^{s_3} = \frac{10+\epsilon}{22+\epsilon}$$

and

$$\lim_{\epsilon \to 0^+} C_3(8-\epsilon, 22, \epsilon) = 39.\overline{09},$$

while we have $C_3(8,22,0) = 34.\overline{54}$.

5.5 Monotonicity, existence, uniqueness and convexity.

Monotonicity is an important property of functions that is required to establish the convergence of most iterative algorithms for solving equilibrium problems modelled as variational inequalities. A function F from a convex subset X of R^n into R^n is monotone on X if, for every x and y in X, there holds

$$\langle F(x) - F(y), x - y \rangle \geq 0.$$

In general, the function C is not monotone. Consider the situation illustrated in Figure 8, together with the strategies $s_1 = ([2,4],[5,4],[\ ],[5],[\ ])$ and $s_2 = ([3],[5,4],[2],[5],[\ ])$ and the strategic flow vectors of the form $x = (x_1, x_2)$. Since both strategies have the same preference order at the unique node where they compete, the single file and parallel loadings are identical. Let $x^1 = (4,6)$. Then $\pi_{12}^{s_1}(x) = 1$ and we have

$$\begin{aligned} C^{s_1}(x) &= 3 + \tfrac{1}{2} \times 3 + \tfrac{1}{2} \times 4 &&= 6\tfrac{1}{2} \\ C^{s_2}(x) &= 3 + \tfrac{1}{2} \times 4 + \tfrac{1}{2} \times 4 &&= 7\tfrac{1}{2}. \end{aligned}$$

If $x^2 = (8,2)$ we obtain

$$\begin{aligned} C^{s_1}(x) &= \tfrac{3}{8} \times a + \tfrac{5}{8}\left(3 + \tfrac{5}{7} \times 3 + \tfrac{2}{7} \times 4\right) &&= \tfrac{21a+220}{56} \\ C^{s_2}(x) &= 4 + \tfrac{5}{7} \times 3 + \tfrac{2}{7} \times 4 &&= \tfrac{51}{7}. \end{aligned}$$

Now

$$\langle C(x^1) - C(x^2), x^1 - x^2 \rangle = \frac{1}{56}\langle (-4,4), (144 - 21a, 12) \rangle,$$

which is negative if $a < 132/21$. This shows that the cost function C is *not* monotone.

Under the assumption that the degenerate case does not occur, existence of at least one solution follows from the continuity of the mapping C and the compactness of the feasible set X. As in linear programming, multiple solutions can coexist and form a convex set. In the strategic model, no counterexample to the convexity of the solution set could be found, although the cost function C may fail to be monotone.

5.6 Monotonicity of the preference order

In order to compute efficiently the best strategic response to a given strategic flow vector x and conditional access probabilities $\pi(x)$ one must, at each node of the transportation network, determine an optimal preference order for each strategy[3] . The optimal order E_j^s should minimize the expression

$$\sum_{k \in E_j^s} \pi_{jk}^s (c_{jk} + C_k^s).$$

[3]Actually, the strategies having identical destination nodes will have identical preference orders E_j^s.

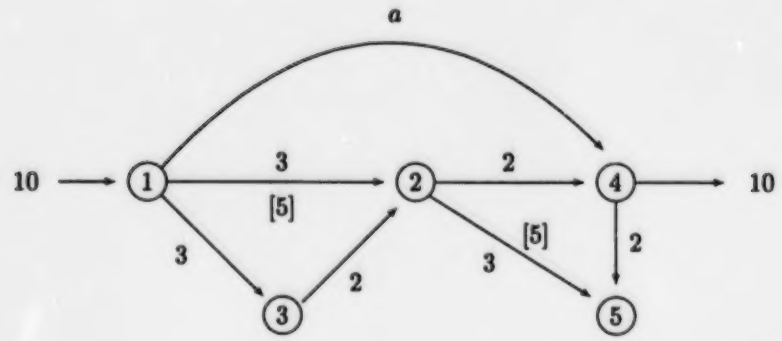

Figure 8: A nonmonotone cost function

A natural way to perform this task is to sort the nodes in the forward star of node j upwards with respect to the criterion $c_{jk_i} + C^s_{k_i}$. This will yield

$$E^{\bar{s}}_j = [k_1, k_2, \ldots, k_l]$$

where

$$c_{jk_i} + C^s_{k_i} \leq c_{jk_{i+1}} + C^s_{k_{i+1}}, \qquad i = 1, 2, \ldots, |j^+| - 1. \tag{3}$$

The next proposition gives an answer to the optimality of these preference sets.

Proposition 2 *In an optimal strategic answer to a given strategic flow vector x and its associated conditional access probability vectors $\pi(x)$, let the preference sets E^s_j be ordered upwards with respect to the quantities $\{c_{jk_i} + C^s_k\}_{k \in j^+}$. This choice is optimal in the single file case but may fail to be optimal in the parallel case.*

Proof In the single file case, let us consider an arbitrary preference order

$$E^s_j = [k_1, k_2, \ldots, k_l],$$

with $l = |j^+|$. With each node $k \in j^+$ we associate a *weight* $e_k = c_{jk} + C^s_k$. We define an *adjacent switch* as the permutation of two adjacent nodes in E^s_j. The adjacent switch of nodes $E^s_j(i)$ and $E^s_j(i+1)$ is *increasing* if the weight of the first node is greater or equal to that of the second node. We will show that an increasing adjacent switch on E^s_j can only decrease the value of C^s_j.

Let us denote by $E^{\tilde{s}}_j$ the preference order resulting from an increasing adjacent switch performed on an index i of E^s_j. We have:

$$E^{\tilde{s}}_j = [k_1, k_2, \ldots, k_{i-1}, k_i, k_{i+1}, \ldots, k_l].$$

Let us summarize the micro-loading phase (loading of a zero flow associated with the optimal strategy currently built) of strategies s and $\tilde{s}$ at node j. After arc (j, k_{i-1}) gets saturated, an *equal* proportion β of the residual flows of strategies s and $\tilde{s}$ is assigned to nodes k_i and k_{i+1}, respectively. At this point, both strategies share the same residual flow proportion ρ. We will consider three cases:

Case 1 Arc (j,k_i) gets saturated first.

When arc (j,k_i) becomes saturated, an equal proportion β of the residual flow of s and $\bar{s}$ is assigned to nodes k_i and k_{i+1} and the node k_i is deleted from the residual preference order of strategy s. At this point, strategies s and $\bar{s}$ have the same residual preference order and residual flow proportion $\rho' = \rho(1-\beta)$. It follows that the micro-loading phase ends identically for both strategies. We denote by β' the proportion of each strategy that accesses node k_{i+1}.

Case 2 Arc (j,k_{i+1}) gets saturated first.

The analysis is symmetric with respect to the previous case. We let $\overline{\beta}$ denote the proportion of residual flow from s and $\bar{s}$ assigned to nodes k_i and k_{i+1} before arc (j,k_{i+1}) gets saturated, and we let $\overline{\beta}'$ denote the proportion of residual flow assigned to node k_i.

Case 3 No arc gets saturated.

This constitutes a limiting case of Case 1 where $\beta = 1$.

The results of the micro-loading of s and $\bar{s}$ are displayed in Table 10.

Case 1

	(j,k_i)	(j,k_{i+1})
s	$\beta\rho$	$\beta'\rho'$
$\bar{s}$	0	$\overline{\beta}\rho + \overline{\beta}'\rho'$

Case 2

	(j,k_i)	(j,k_{i+1})
s	$\beta\rho + \beta'\rho'$	0
$\bar{s}$	$\overline{\beta}'\rho'$	$\overline{\beta}\rho$

Case 3

	(j,k_i)	(j,k_{i+1})
s	ρ	0
$\bar{s}$	0	ρ

Table 10: Conditional access probabilities to arcs (j,k_i) and (j,k_{i+1}) for s and $\bar{s}$

As a general rule we obtain the relationships

$$\begin{aligned} \pi^s_{jk_i} + \pi^s_{jk_{i+1}} &= \pi^{\bar{s}}_{jk_i} + \pi^{\bar{s}}_{jk_{i+1}} \\ \pi^{\bar{s}}_{jk_{i+1}} &\geq \pi^s_{jk_{i+1}}. \end{aligned}$$

from which we deduce

$$C^s_j - C^{\bar{s}}_j = (e_{k_{i+1}} - e_{k_i})(\pi^s_{jk_{i+1}} - \pi^{\bar{s}}_{jk_{i+1}}) \geq 0.$$

Thus an increasing adjacent switch does not deteriorate the preference order E^s_j. Let $E^{s^*}_j$ be an optimal preference order. After a finite number of increasing adjacent switches, we obtain the preference order $E^{\bar{s}}_j$, which must therefore be optimal.

6 Solution algorithms

6.1 General considerations

The quest for a strategic equilibrium comes down to finding a vector x^* that solves the variational inequality IV(C,X), i.e.,

$$\langle C(x^*), x^* - x \rangle \leq 0 \qquad \forall x \in X.$$

The challenge in solving this mathematical program is threefold:

- In contrast with uncapacitated case, strategic flows, of which there is an exponential number, have to be kept explicitly. This is due to the fact that arc flows are *not* a linear function of strategic flows;
- The cost function is not available in closed form. This rules out methods using nontrivial linesearch techniques;
- The cost function cannot be assumed to be either integrable (a gradient mapping), differentiable nor monotone.

The first 'non property' reduces the algorithmic choice: one has to adopt some sort of restriction strategy. Within the restriction framework, several methods can be developed. We will present in this algorithmic variations on either a 'Frank and Wolfe' or a projective theme. Since the cost function C is nonmonotone, convergence of the iterates towards an equilibrium solution x^* is not guaranteed. In this context, the algorithms must be viewed as natural but heuristic procedures.

6.2 A stopping criterion

The algorithms are halted whenever some criterion is met. For variational inequalities, so-call 'gap functions' provide a computable criterion. A nonnegative function g is a gap function if its zeroes coincide with the solutions of the variational inequality. A natural gap function is defined as

$$g_P(x) = \max_{y \in X} \langle F(x), x - y \rangle, \tag{4}$$

which is a byproduct of the computation of a best strategy via the CAPSHORT procedure. In this work we make use of regularized, or relative gap function , which is simply a scaled version of g_P by the total transportation cost:

$$g_R(x) = \frac{g_P(x)}{\langle S(x), x \rangle}. \tag{5}$$

The procedure that evaluates the gap function is listed below.

PROCEDURE GAP (x)

input: $x = \{x_s\}_{s \in W}$ *[strategic flow vector]*
output: $g_R(x)$ *[value of relative gap at x]*
$\overline{x}$ *[extremal solution of $\min_{z \in X} \langle C(x), y \rangle$]*

INITIALIZATION
for $s \in W$ **do**
  $\overline{x}_s \leftarrow 0$ *[optimal flow assigned to strategy s]*
endfor
$C^T = 0$ *[total cost $\langle C(x), x \rangle$]*
$C^* = 0$ *[optimal value of linear subproblem]*

COMPUTATION OF TOTAL COST
CAPLOAD$(x) \rightarrow z$, C *[loading of x]*
for $s \in W$ **do**
  $C^T \leftarrow C^T + x_s C^s$ *[total cost update]*
endfor

COMPUTATION OF OPTIMAL TOTAL COST
for $(q,r) \in L$ **do**
  CAPSHORT$(z,(q,r)) \rightarrow \overline{s}$, $C^{\overline{s}}$ *[computing an optimal strategy]*
  $C^* \leftarrow C^* + \overline{x}_{\overline{s}} C^{\overline{s}}$ *[optimal total cost update]*
endfor

$g_R(x) \leftarrow (C^T - C^*)/C^T$ *[gap evaluation]*

Note that in the actual implementation, the procedure CAPSHORT was called once for each origin node, and not for each origin-destination pair.

6.3 A Frank and Wolfe approach

The popularity of the Frank and Wolfe (FW) algorithm [4] for solving traffic assignment problems is due to its efficient handling of network structures: at each iteration, shortest paths have to be computed for each origin (or destination). Each iteration of the FW method for minimizing a differentiable function f over a compact,convex set X takes the form

$$x^{k+1} = x^k + \theta_k(\overline{x}^k - x^k),$$

where $\overline{x}^k \in \arg\min_{y \in X} \langle \nabla f(x^k), y \rangle$ and $\theta_k \in [0,1]$ is some suitable stepsize. A popular choice for the stepsize θ_k is the harmonic sequence $1/(k+1)$. We introduce a set W that contains all strategies generated by the procedure.

STRATEQ1

input: strategic flow vector x, tolerance factor ϵ
output: strategic equilibrium x^*

INITIALIZATION
GAP $(x) \rightarrow \overline{x}$, $g(x)$
$k \leftarrow 1$

MAIN LOOP
while $g(x) \geq \epsilon$ **do**
$x \leftarrow (1-\theta^k)x + \theta^k \overline{x}$
GAP (x)
$k \leftarrow k+1$
$W \leftarrow W \cup \{\overline{x}\}$
endwhile

The initial vector x is computed with respect to free-flow travel times. Although the cost of its origin-destination components is equal to that of shortest O-D paths, nevertheless it contains the relevant strategic informations at nodes of the network since it is produced by the CAPSHORT procedure.

In order to the limit the size of the working set W, we adopt a restriction strategy which is reminiscent of the RSD (Restricted Simplicial Decomposition) technique of Lawphongpahnich and Hearn [9]. At every iteration, an optimal strategy $\overline{s}$ is corresponding to some OD couple is included in the working set only if its cost (with respect to the current strategic vector x) is significantly inferior to that of the current strategy $\tilde{s}$, i.e.,

$$C^{\overline{s}} \leq C^{\tilde{s}} - \epsilon_1,$$

for some predetermined quantity ϵ_1. Moreover, any strategy whose flow value falls below some threshold value ϵ_2 is deleted from the working set.

A drawback of the Frank-Wolfe method is the averaging aggregation process that takes place over all origin-destination pairs: adopting a common stepsize may actually worsen the equilibrium conditions for some O-D pairs and have an adverse effect on convergence. To remedy this situation, one may use different stepsizes for each O-D pair. The stepsize rule that we adopted for O-D pair k is

$$\theta_s^k = 1 - \frac{C^{\overline{s}}}{C^s}.$$

The quantities θ_s^k are adaptive steps that measure the degree of insatisfaction of the users travelling from the origin node to the destination node of the O-D pair k. Note that the implementation of this idea does not incur much extra computational work since the strategic information is readily available in disaggregate form.

6.4 Projection algorithms

A vector x^* is solution of the variational inequality IV(F,X) if and only if it is a solution of the fixed point problem

$$x^* = p_\alpha(x^*),$$

where α is a positive scalar and $p_\alpha(x) = \text{proj}_X(x - \alpha F(x))$ represents the Euclidian projection of the vector $x - \alpha F(x)$ over the convex set X. In our situation, the set X can be defined as the convex combination of strategic flows in the working set W, i.e., the Euclidian product of simplices defined defined by nonnegativity and demand constraints. The "vanilla style" projection algorithm generates a sequence of iterates according to the formula

$$x^{k+1} = p_\alpha(x^k) = \text{proj}_X(x^k - \alpha F(x^k),$$

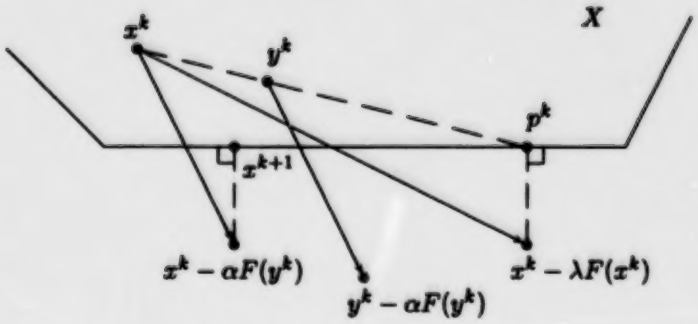

Figure 9: Konnov's algorithm

where the projection over the simplex is a very simple operation.

Several variants of this basic scheme have been proposed over the years. One of them, due to Konnov [6] converges under weaker conditions[4] that the projection algorithm and performs automatic updates of its internal parameters. For specific choices of these parameters, Konnov's algorithm reduces to the robust extragradient method of Korpelevich [7].

Konnov's algorithm makes use of positive parameters α, λ and $\theta \in (0,1)$. The computation of x^{k+1} requires two projections and one convex combination. The computational cost of these operations is very small with respect to the CAPLOAD and CAPSHORT steps. The algorithm is defined by the following three equations:

$$\begin{aligned} p^k &= \text{proj}_X(x^k - \lambda F(x^k)), \\ y^k &= (1-\theta)x^k + \theta p^k, \\ x^{k+1} &= \text{proj}_X(x^k - \lambda F(y^k)) \end{aligned}$$

and is illustrated geometrically in Figure 9.

The projection methods have been combined with the pseudo Frank and Wolfe (PFW) method as follows: the iterations are divided into cycles of iterations. Each cycle is made of $K = K_1 + K_2$ iterations, where the K_1 first iterations are projection iterations and the remaining K_2 are PFW iterations destined to enrich the working set W.

7 Numerical results

The numerical tests have been designed to show the applicability of our method and to illustrate the convergence behaviour of the algorithms that we implemented. They have been performed on a small example where the equilibrium is known and and on a larger networks where restriction strategies had to be implemented. Numerical results are provided for both the single file and parallel processings. The quality of an approximate solution is measured using the relative gap function 5.

[4]Konnov's scheme is globally convergent if the cost function C is pseudomonotone over the definition set X. In the single file case, we could not find counterexamples to the pseudomonotonicity of C.

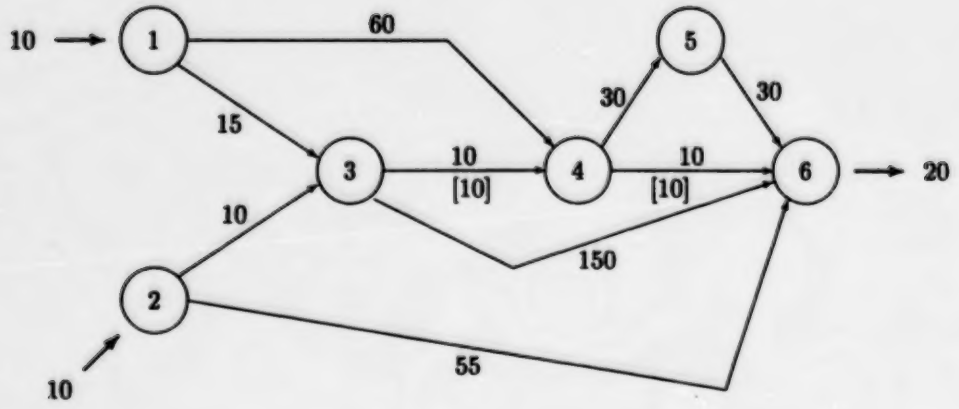

Figure 10: A small test network

7.1 A small network

Let us consider the network of Figure 10, inspired from a transit assignment situation where links (3,4) and (4,6) are served by two bus lines with equal capacity of ten seats. The strategies of interest are

node :	1	2	3	4	5	6
s_1 :	[3]	[ ]	[4,6]	[6,5]	[6]	[ ]
s_2 :	[4]	[ ]	[ ]	[6,5]	[6]	[ ]
s_3 :	[ ]	[3]	[4,6]	[6,5]	[6]	[ ]
s_4 :	[ ]	[6]	[ ]	[ ]	[ ]	[ ].

It can be argued that strategy s_3 carries positive flow at equilibrium. Otherwise, its cost (35) would be less that the expected cost of strategy s_4 (55). This shows that, at equilibrium, the (expected) cost of s_3 is at most 55. Since the cost of strategy s_2 exceeds the cost of strategy s_3 by five units, we deduce that, at equilibrium, it must be equal to 55+5=60, which is less than the cost of strategy s_1. Hence, all flow originating from node 1 will use strategy s_1. Now, if all flow from origin 2 used s_3, the cost of this strategy would be equal to

$$C_3^* = 10 + \frac{10}{20} \times (10 + 10) + \frac{10}{20} \times 150 = 95\,.$$

clearly a non-equilibrium situation. We conclude from these observations that x_3 and x_4 are positive. To find the value of x_3, we simply write that its cost must be equal to 55, i.e.,

$$10 + \frac{10}{10 + x_3^*}(10 + 10) + \frac{x_3^*}{10 + x_3^*}150 = 55\,.$$

The solution of this linear equation being $x_3^* = 50/21$, we obtain the unique equilibrium strategic flow $x^* = (10, 0, 50/21, 160/21)$, for both the single file and parallel cases. In the vicinity of the equilibrium solution $x^* = (10, 0, 50/21, 160/21)$, the cost function C takes the form

$$\begin{aligned} C_1(x) &= 15 + \left(\frac{10}{x_1 + x_3}\right)\left(10 + \frac{10}{10 + x_2}(10) + \frac{x_2}{10 + x_2}(60)\right) + \left(\frac{x_1 + x_3 - 10}{x_1 + x_3}\right)150 \\ C_2(x) &= 60 + \left(\frac{10}{10 + x_2}\right)10 + \left(\frac{x_2}{10 + x_2}\right)60 \end{aligned}$$

# iter.	Strategic flows				Strategic costs				Relative
	s_1	s_2	s_3	s_4	s_1	s_2	s_3	s_4	gap (%)
0	10.00	0.00	10.00	0.00	100.00	70.00	95.00	55.00	35.897
1	5.00	5.00	5.00	5.00	51.67	86.67	46.67	55.00	18.056
2	6.67	3.33	6.67	3.33	76.88	82.50	71.88	55.00	9.052
3	7.50	2.50	5.00	5.00	69.00	80.00	64.00	55.00	5.524
5	8.33	1.67	3.33	6.67	59.69	77.14	54.69	55.00	2.649
10	9.09	0.91	2.73	7.27	58.53	74.17	53.53	55.00	2.177
20	9.52	0.48	2.86	7.14	61.84	72.27	56.84	55.00	0.867
50	9.80	0.20	2.55	7.45	60.54	70.96	55.54	55.00	0.295
100	9.90	0.10	2.48	7.52	60.36	70.49	55.36	55.00	0.163
200	9.95	0.05	2.44	7.56	60.26	70.25	55.26	55.00	0.098
500	9.98	0.02	2.40	7.60	60.03	70.10	55.03	55.00	0.024
1000	9.99	0.01	2.39	7.61	60.01	70.05	55.01	55.00	0.011
2000	10.00	0.00	2.38	7.62	60.00	70.02	55.00	55.00	0.005
10000	10.00	0.00	2.38	7.62	60.00	70.00	55.00	55.00	0.001

Table 11: **STRATEQ1** on the small network

$$\begin{aligned} C_3(x) &= C_1(x) - 5 \\ C_4(x) &= 55. \end{aligned}$$

The Jacobian matrix of C at the solution x^* is

$$J = C'(x^*) = \begin{pmatrix} \frac{441}{52} & \frac{105}{26} & \frac{441}{52} & 0 \\ 0 & 5 & 0 & 0 \\ \frac{441}{52} & \frac{105}{26} & \frac{441}{52} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Since the eigenvalues of $J + J^t$ are nonnegative, the cost function C is monotone at x^*. This property might partially explain the good behaviour of the solution algorithms.

The convergence of the **STRATEQ1** method (Table 11), based on harmonic and global stepsizes, is quite similar to that of Frank and Wolfe's algorithm in an uncapacitated case. The adaptive stepsize strategy **STRATEQ2** (Table 12), on the other hand, converged very rapidly to the equilibrium solution. As expected, the projective methods, under standard choices of their respective parameters, converged yet faster. The results are presented in Tables 13, 14 and 15. From Table 16, the first two variants of the projection algorithms come up first. Konnov's algorithm might be preferred to the vanilla style projection method on the basis of its better theoretical convergence properties.

8 A larger instance: the Sioux Falls network

The second set of tests was performed on the network illustrated in Figure 11. This network of 24 nodes, 41 nodes and 4 OD couples (see Table 17) is a simplified acyclic version of the network used by Suwansirikul [15] in the static case. Although this network is small, the complexity of its strategic structure can be observed on the strategy shown in Table 18.

On this example the tolerance parameters ϵ_1 and ϵ_2 have been set to 10^{-4} and .005 for algorithm **STRATEQ1** and to 10^{-4} and 10^{-4} for **STRATEQ2**. The projective methods were initialized by 100 iterations of the PFW algorithm. Next the method proceed along through 20 cycles, each cycle consisting of

# iter.	Strategic flows				Strategic costs				Relative
	s_1	s_2	s_3	s_4	s_1	s_2	s_3	s_4	gap (%)
0	10.00	0.00	10.00	0.00	100.00	70.00	95.00	55.00	35.897
1	7.00	3.00	5.79	4.21	72.38	81.54	67.38	55.00	7.221
2	7.34	2.66	4.73	5.27	65.95	80.51	60.95	55.00	5.242
3	7.82	2.18	4.26	5.74	64.82	78.95	59.82	55.00	4.113
5	8.52	1.48	3.65	6.35	63.48	76.44	58.48	55.00	2.619
10	9.39	0.61	2.90	7.10	61.57	72.87	56.57	55.00	0.970
20	9.88	0.12	2.48	7.52	60.33	70.60	55.33	55.00	0.179
50	10.00	0.00	2.38	7.62	60.00	70.01	55.00	55.00	0.002
100	10.00	0.00	2.38	7.62	60.00	70.00	55.00	55.00	0.000

Table 12: **STRATEQ2** on small network

# iter.	Strategic flows				Strategic costs				Relative
	s_1	s_2	s_3	s_4	s_1	s_2	s_3	s_4	gap (%)
0	10.00	0.00	10.00	0.00	100.00	70.00	95.00	55.00	35.897
1	7.00	3.00	6.00	4.00	73.88	81.54	68.88	55.00	7.616
2	7.77	2.23	4.61	5.39	67.36	79.13	62.36	55.00	4.691
3	8.94	1.06	3.88	6.12	67.33	74.78	62.33	55.00	2.880
5	10.00	0.00	2.66	7.34	62.29	70.00	57.29	55.00	0.516
10	10.00	0.00	2.38	7.62	60.00	70.00	55.00	55.00	0.000

Table 13: Projection algorithm on small network ($\alpha = 0.2$)

# iter.	Strategic flows				Strategic costs				Relative
	s_1	s_2	s_3	s_4	s_1	s_2	s_3	s_4	gap (%)
0	0.00	10.00	5.00	5.00	51.67	86.67	46.67	55.00	28.485
1	3.50	6.50	5.83	4.17	53.42	88.42	48.42	55.00	20.019
2	7.00	3.00	6.49	3.51	77.19	81.54	72.19	55.00	8.616
3	7.43	2.57	4.77	5.23	66.87	80.21	61.87	55.00	5.211
5	9.50	0.50	3.27	6.73	65.10	72.36	60.10	55.00	1.662
10	10.00	0.00	2.38	7.62	60.00	70.00	55.00	55.00	0.000

Table 14: Konnov's algorithm on small network ($\alpha = 0.2$, $\lambda = 0.1$ et $\theta = 0.001$)

# iter.	Strategic flows				Strategic costs				Relative
	s_1	s_2	s_3	s_4	s_1	s_2	s_3	s_4	gap (%)
0	2.00	8.00	6.00	4.00	53.75	88.75	48.75	55.00	22.932
1	3.72	6.28	6.25	3.75	54.24	89.24	49.24	55.00	18.925
2	4.51	5.49	5.77	4.23	55.75	87.72	50.75	55.00	15.380
3	5.26	4.74	5.32	4.68	57.26	86.08	52.26	55.00	12.018
5	6.63	3.37	4.53	5.47	59.81	82.60	54.81	55.00	6.357
10	9.19	0.81	3.13	6.87	62.48	73.75	57.48	55.00	1.417
20	10.00	0.00	2.42	7.58	60.32	70.00	55.32	55.00	0.067
50	10.00	0.00	2.38	7.62	60.00	70.00	55.00	55.00	0.000

Table 15: Extragradient algorithm on small network ($\alpha = 0.1$)

Method	gap function (%)									
	10^{1}	10^{0}	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
STRATEQ1	2	13	106	883	8716	–	–	–	–	–
STRATEQ2	1	10	24	39	54	69	84	99	104	109
Projection	1	5	6	8	9	10	11	13	14	15
Konnov	2	6	7	9	10	11	12	14	15	16
Extragradient	4	11	19	27	35	44	52	60	68	77

Table 16: Number of iterations required to achieve a 10^{-k} gap value(small network)

O-D pair	Notation	Demand
(1,24)	**OD1**	35
(1,22)	**OD2**	25
(7,24)	**OD3**	20
(7,22)	**OD4**	20

Table 17: OD information for the Sioux Falls network

node	1	2	3	4	5	6	7	8
preference set	[3]	[ ]	[4]	[11,5]	[9,6]	[8]	[ ]	[16,9]

node	9	10	11	12	13	14	15	16
preference set	[10]	[11,15,16,17]	[14,12]	[13]	[24]	[23,15]	[20,17,19]	[17, 18]

node	17	18	19	20	21	22	23	24
preference set	[19]	[22]	[20]	[23,22,21]	[24]	[24]	[24]	[ ]

Table 18: A typical strategy for the Sioux Falls network (OD pair 1-24)

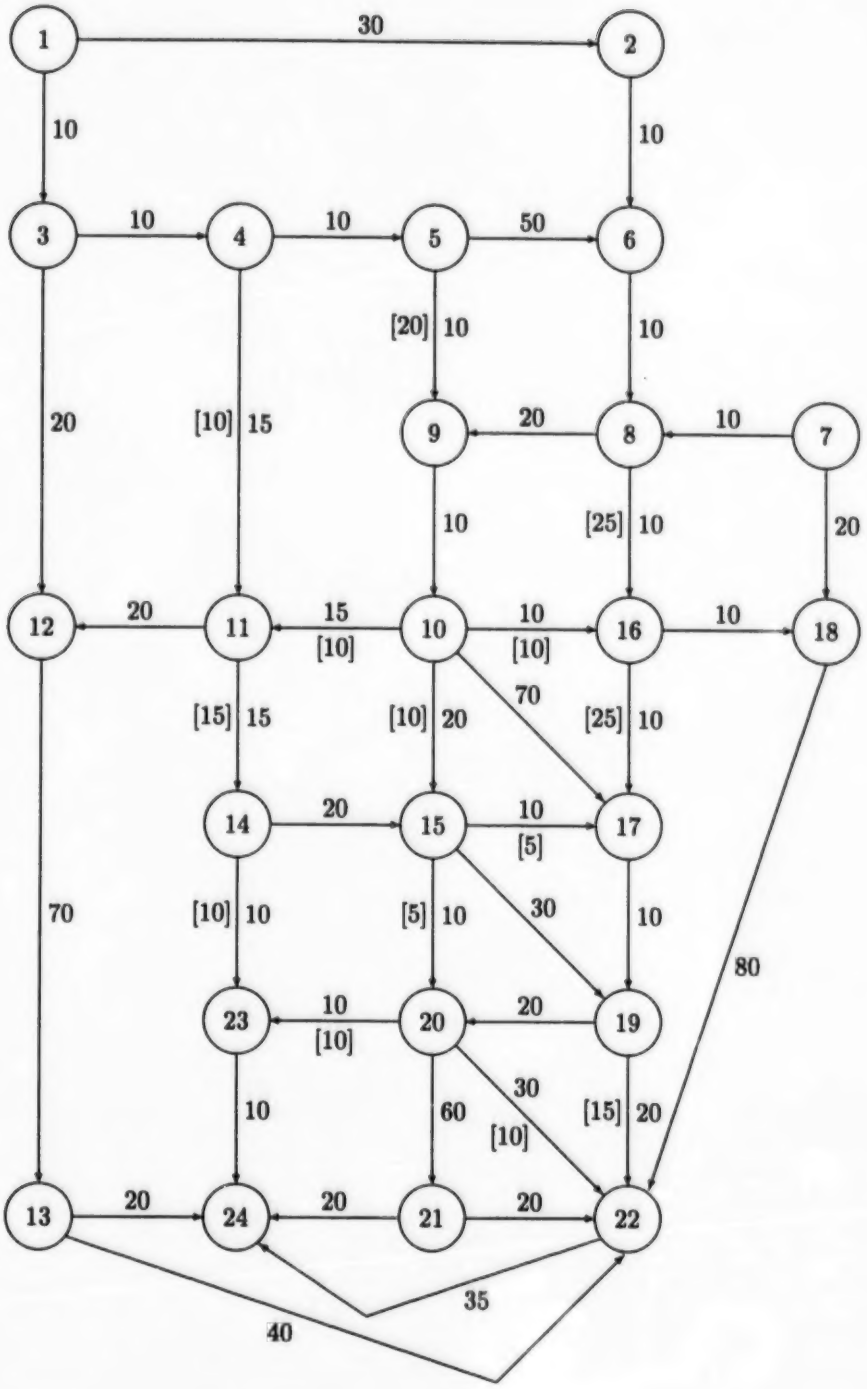

Figure 11: The Sioux Falls network

- 100 projection steps over the set of strategies in the current working set;
- 5 PFW steps aimed at enriching the working set.

In the parallel case, numerical tests were conducted with both the exact and heuristic procedures for building the current best strategic flow vector. The exact method is only practical when the network under consideration has low density, which is the case of the Sioux Falls network. If the heuristic is adopted, one must realize that the estimated gap is no more a reliable measure of equilibrium and can even turn out to be negative!

The numerical results for PFW are displayed in Tables 19 and 20. We give, for each OD pair, the number of strategies considered and the contribution of each OD to the gap function. Table 21 makes comparisons with solutions obtained by PFW after 10 000 iterations. The results for the parallel case are displayed in Tables 23, 24, 25 and 26 for the exact case, and Tables 27 and 28) for the heuristic case. An asterisk (*) is used to denote the tests based on the parallel processing.

The performance of the algorithms is satisfactory, especially given the complexity of the cost function C. In the single file case, about 20 iterations are sufficient to bring the gap function below 1%. Under the adaptive stepsize rule **STRATEQ2**, quasi-optimal solutions were obtained. The best asymptotic performance was achieved by the combination of PFW and Konnov's double projection method. We note that the size of the working is larger with **STRATEQ1** than with **STRATEQ2**. More important, all methods converge towards the same strategic flow vector. A 'microscopic' analysis of the results actually revealed no significant differences between the solutions. In the parallel case, the asymptotic behaviour is not good under the heuristic rule. This was, of course, to be expected.

The numerical tests confirmed that the single file and parallel processings influence significantly the solution, in terms of both equilibrium strategies and costs. We observed that the number of strategies involved in the parallel case was higher.

Although we exhibited counterexamples for the monotonicity property, we found it interesting to check whether this condition actually held for our test problems. In Table 29 one will find counterexample for both the single file and parallel cases. Moreover we found a counterexample to the pseudomonotonicity[5] of C for the parallel case. This is an important negative result since the proof of global convergence for the extragradient and Konnov methods are based on the pseudomonotonicity of the cost function.

9 Random grid networks

The last set of numerical tests has been performed on rectangular grid networks with randomly generated costs. A grid is composed of mn nodes and $2mn - m - n$ arcs. The horizontal arcs are oriented from left to right and the vertical arcs are oriented either upwards or downwards, depending whether they lie on odd-numbered or even-numbered columns of the grid (see Figure 12). Such a network can trivially be topologically ordered.

A number l of OD pairs is randomly selected on the grid. In order to be assured that there is at least one feasible path from an origin to its corresponding destination

- the column of the destination should lie to the right of the origin's column;
- the origin and the corresponding destination should lie on distinct columns.

Demand is uniformly distributed over all OD pairs and arc costs follow a uniform distribution $\mathcal{U}[10, 100]$. The capacity of the arcs on the Hamiltonian path starting at the southwestern node is set to $+\infty$, thus

[5] A function F is pseudomonotone on a convex set X if, for any x and y in X, we have: $\langle F(x), x-y\rangle \leq 0 \Rightarrow \langle F(y), x-y\rangle \leq 0$. A monotone function is pseudomonotone but the converse statement does not hold in general.

	Number of strategies				Gap contribution				
	OD1	**OD2**	**OD3**	**OD4**	**OD1**	**OD2**	**OD3**	**OD4**	total
0	1	1	1	1	10.984	5.749	2.833	4.599	24.165
1	2	2	2	2	4.378	4.410	3.915	1.855	14.559
2	3	3	3	3	2.460	0.905	2.673	0.647	6.685
3	3	3	4	3	0.137	1.537	1.416	0.099	3.189
5	4	4	4	4	0.475	1.538	1.014	0.238	3.264
10	4	5	4	5	1.120	0.134	0.606	0.121	1.980
20	4	5	5	5	0.080	0.316	0.306	0.161	0.863
50	4	6	5	5	0.184	0.119	0.124	0.073	0.500
100	4	7	5	5	0.087	0.102	0.066	0.049	0.304
200	4	7	5	5	0.020	0.018	0.035	0.011	0.085
500	4	6	2	4	0.016	0.011	0.002	0.007	0.036
1000	4	5	2	2	0.017	0.008	0.005	0.006	0.036
2000	2	5	2	2	0.005	0.001	0.003	0.001	0.011
10000	2	4	2	2	0.000	0.000	0.001	0.000	0.002

Table 19: **STRATEQ1** on Sioux Falls network

	Number of strategies				Gap contribution				
	OD1	**OD2**	**OD3**	**OD4**	**OD1**	**OD2**	**OD3**	**OD4**	total
0	1	1	1	1	10.984	5.749	2.833	4.599	24.165
1	2	2	2	2	3.680	1.791	0.886	1.321	7.677
2	2	2	3	2	1.632	2.069	0.821	0.409	4.932
3	2	3	3	2	0.905	2.090	0.711	0.233	3.939
5	2	3	3	2	0.393	1.735	0.560	0.142	2.830
10	2	3	3	2	0.116	0.950	0.347	0.105	1.518
20	3	3	3	2	0.068	0.250	0.133	0.068	0.520
50	3	4	3	3	0.016	0.051	0.014	0.030	0.111
100	3	5	2	3	0.007	0.030	0.014	0.028	0.080
200	3	5	2	3	0.005	0.018	0.013	0.018	0.054
500	3	5	2	3	0.003	0.011	0.007	0.007	0.028
1000	3	5	2	3	0.002	0.007	0.003	0.003	0.014
2000	3	5	2	3	0.000	0.003	0.001	0.001	0.004
10000	2	5	1	2	0.000	0.000	0.000	0.000	0.000

Table 20: **STRATEQ2** on Sioux Falls network

Heuristic	Minimum strategic cost				Gap (%)
	OD1	**OD2**	**OD3**	**OD4**	
STRATEQ1	120.00	139.99	111.96	100.00	1.6×10^{-3}
STRATEQ2	120.00	140.00	111.81	100.00	1.8×10^{-7}

Table 21: Solutions after 10 000 iterations (Sioux Falls)

# cycles	Gap (%)		
	Projection ($\alpha = 0.3$)	Konnov ($\alpha = 0.4$, $\lambda = 0.1$, $\theta = 0.1$)	Extragradient ($\alpha = 0.2$)
1	1.4×10^{-2}	1.5×10^{-2}	1.4×10^{-2}
5	1.4×10^{-2}	1.3×10^{-3}	1.5×10^{-2}
10	6.9×10^{-3}	9.9×10^{-4}	1.3×10^{-2}
15	4.1×10^{-4}	4.2×10^{-5}	6.3×10^{-3}
20	4.0×10^{-5}	2.0×10^{-6}	8.8×10^{-4}

Table 22: Combined FW-Projection under **STRATEQ2** (Sioux Falls)

	Number of strategies				Gap contribution				
	OD1	**OD2**	**OD3**	**OD4**	**OD1**	**OD2**	**OD3**	**OD4**	total
0	1	1	1	1	10.840	5.463	5.125	4.370	25.798
1	2	2	2	2	0.879	3.357	2.048	1.518	7.802
2	2	3	3	2	3.123	3.823	0.599	0.643	8.188
3	2	4	3	3	2.048	1.447	1.231	0.238	4.963
5	3	5	3	4	0.893	0.513	0.569	0.163	2.139
10	3	5	3	4	1.074	0.450	0.314	0.071	1.909
20	4	7	5	5	0.225	0.140	0.108	0.068	0.540
50	4	8	5	5	0.179	0.119	0.068	0.036	0.402
100	4	11	5	5	0.058	0.049	0.033	0.009	0.149
200	4	12	5	5	0.020	0.022	0.019	0.007	0.068
500	4	13	4	5	0.005	0.025	0.004	0.012	0.046
1000	4	11	4	5	0.021	0.029	0.007	0.007	0.064
2000	4	11	4	5	0.001	0.005	0.001	0.002	0.008
10000	3	9	4	4	0.001	0.001	0.000	0.000	0.003

Table 23: **STRATEQ1** applied to Sioux Falls network (*)

	Number of strategies				Gap contribution				
	OD1	**OD2**	**OD3**	**OD4**	**OD1**	**OD2**	**OD3**	**OD4**	total
0	1	1	1	1	10.840	5.463	5.125	4.370	25.798
1	2	2	2	2	5.237	1.348	2.955	0.998	10.537
2	2	2	3	2	2.473	1.586	1.938	0.399	6.397
3	3	3	3	2	2.503	1.379	1.685	0.332	5.900
5	3	5	3	2	1.460	1.349	1.109	0.213	4.131
10	3	7	3	2	0.509	0.982	0.490	0.221	2.201
20	3	7	3	2	0.239	0.302	0.196	0.111	0.849
50	4	8	4	3	0.046	0.058	0.073	0.005	0.181
100	4	8	4	5	0.019	0.008	0.038	0.005	0.070
200	4	8	3	5	0.008	0.003	0.014	0.002	0.027
500	4	10	3	5	0.002	0.003	0.004	0.001	0.009
1000	4	10	3	5	0.001	0.002	0.002	0.001	0.006
2000	4	10	3	5	0.001	0.001	0.001	0.000	0.003
10000	4	10	3	5	0.000	0.000	0.000	0.000	0.000

Table 24: **STRATEQ2** applied to Sioux Falls network (*)

Heuristic	Minimum strategic cost				Gap (%)
	OD1	**OD2**	**OD3**	**OD4**	
STRATEQ1	120.00	140.00	114.30	100.00	2.9×10^{-3}
STRATEQ2	120.00	140.00	114.31	100.00	3.2×10^{-4}

Table 25: Solutions after 10 000 iterations (Sioux Falls) (*)

# cycle	Gap (%)		
	Projection ($\alpha = 0.2$)	Konnov ($\alpha = 0.2$, $\lambda = 0.1$, $\theta = 0.1$)	Extragradient ($\alpha = 0.1$)
1	1.2×10^{-2}	1.2×10^{-2}	1.3×10^{-2}
5	4.8×10^{-3}	4.8×10^{-3}	7.4×10^{-3}
10	2.3×10^{-3}	2.3×10^{-3}	4.6×10^{-3}
15	1.3×10^{-3}	1.3×10^{-3}	3.1×10^{-3}
20	7.1×10^{-4}	7.1×10^{-4}	2.1×10^{-3}

Table 26: Combined FW-Projection under **STRATEQ2** (Sioux Falls) (*)

# iter.		Heuristic gap (%)	True gap (%)
0	4	25.798	25.798
1	8	7.802	7.802
2	10	6.310	6.310
3	14	3.229	3.229
5	15	4.208	4.208
10	15	1.616	1.622
20	16	1.022	1.061
50	16	0.474	0.536
100	16	0.133	0.430
200	16	0.104	0.316
500	16	0.019	0.359
1000	16	0.009	0.333
2000	15	0.004	0.345

Table 27: **STRATEQ1**: heuristic approach on Sioux Falls network (*)

# iter.		Heuristic gap (%)	True gap (%)
0	4	25.798	25.798
1	8	10.537	10.537
2	9	6.397	6.397
3	11	5.725	5.900
5	13	4.327	4.327
10	14	1.988	2.458
20	14	0.933	0.933
50	16	0.214	0.342
100	16	0.074	0.372
200	15	−0.089	0.626
500	15	−0.044	0.663
1000	15	−0.016	0.732
2000	11	−0.001	0.725

Table 28: **STRATEQ2**: heuristic approach on Sioux Falls network (*)

Queue discipline	# tests	Counterexample to		Suboptimality of **CAPSHORT**
		Monotony	pseudomonotonicity	
Single file	1000	24	1	0
Parallel files	1000	1	0	213

Table 29: Influence of queue discipline (Sioux Falls)

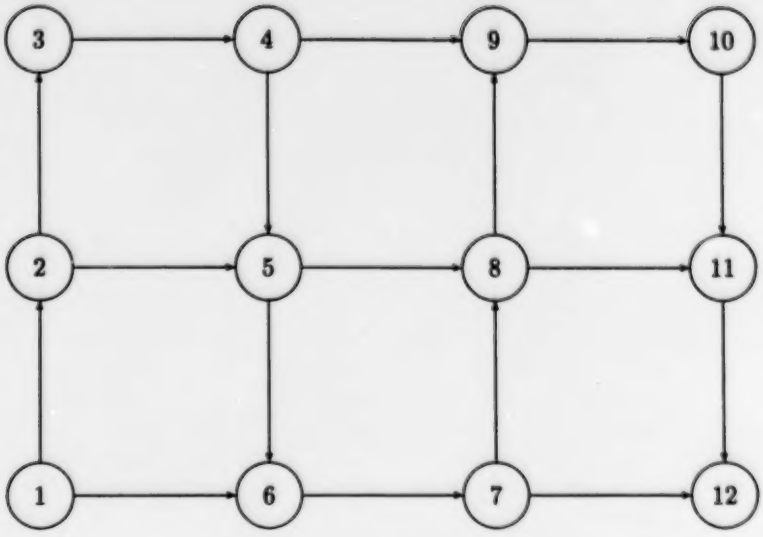

Figure 12: A grid network ($m = 4$ et $n = 3$)

allowing uncapacitated paths for every OD pair. Other arc capacities are randomly distributed according to a uniform distribution $\mathcal{U}[10d/(10mn), 50]$. Table 30 presents the parameters used for each of five grid networks.

Since the outdegree of each arc is (at most) 2, there is no need to distinguish between the single file and parallel cases, according to Proposition 1. The numerical tests were based on the PFW algorithm coupled with the **STRATEQ2** adaptive stepsize strategy. Control parameters ϵ_1 and ϵ_2 were set to 0.1. The tests were performed in the *C* programming language on a *Sun Ultra 10* computer running under the *Solaris* operating system. Results are displayed in Tables 31 to 35. Another statistic of interest is the running time of one main iteration of **STRATEQ2** as a function of the number of active strategies (see Table 36). This was achieved by running PFW with no control on the size of the working set W. In order to obtain a reliable estimate, the running time was averaged over three iterations, for each fixed size of the working set.

On all grid test problems, a relative gap measure of 1% was reached after roughly 20 iterations. The decrease of the gap function on grids **A** and **E** is illustrated in Figure 13. The number of active strategies is, on the average, equal to tow for each OD pair. As expected, the running time of a single iteration, as illustrated in Figure 14, grows linearly with the number of strategies.

Finally, we implemented the standard projection algorithm within an RSD (restricted simplicial decomposition) framework. Each main iteration consists in 5 steps of the projection algorithm followed by one **STRATEQ2** step. The initial strategic flow vector was obtained by performing 5 **STRATEQ2** steps. The results (see Table Projections) show a clear improvement over strategy **STRATEQ2** and lead us to believe that truly large scale instances could be tackled using this algorithmic approach.

	A	B	C	D	E
m	10	20	20	20	40
n	8	8	16	32	32
l	10	10	20	20	40
d	1000	1000	1000	1000	1000

Table 30: Parameters of the grid networks

# iter.	$\|W\|$	gap (%)	Cpu (sec)
0	10	30.70	0.04
1	16	9.23	0.10
2	20	6.91	0.18
3	21	5.61	0.26
5	22	4.10	0.44
10	24	2.35	0.95
20	24	1.07	2.01
40	23	0.35	4.33
60	22	0.15	6.24
80	21	0.08	8.00
100	20	0.05	9.63

Table 31: Numerical results: grid **A**

# iter.	$\|W\|$	gap (%)	Cpu (sec)
0	10	65.11	0.08
1	19	20.04	0.22
2	28	8.49	0.44
3	33	6.56	0.71
5	35	4.61	1.28
10	39	2.56	2.87
20	40	1.15	6.28
40	40	0.39	13.21
60	35	0.19	19.72
80	29	0.12	25.02
100	27	0.09	29.75

Table 32: Numerical results: grid **B**

# iter.	$\|W\|$	gap (%)	Cpu (sec)
0	20	47.45	0.55
1	40	15.16	1.66
2	58	9.55	3.39
3	64	7.80	5.38
5	68	5.67	9.83
10	70	3.09	21.38
20	72	1.26	45.92
40	63	0.37	92.57
60	55	0.17	129.69
80	50	0.10	161.56
100	45	0.06	190.78

Table 33: Numerical results: grid **C**

# iter.	$\|W\|$	gap (%)	Cpu (sec)
0	20	68.87	1.26
1	36	27.75	3.71
2	52	11.25	7.59
3	61	8.53	12.49
5	65	5.70	22.87
10	65	2.73	49.43
20	64	0.95	101.60
40	52	0.25	186.62
60	43	0.12	252.32
80	40	0.09	308.99
100	38	0.07	361.00

Table 34: Numerical results: grid **D**

# iter.	$\|W\|$	gap (%)	Cpu (sec)
0	40	80.06	8.17
1	76	20.69	26.07
2	111	9.56	57.48
3	132	7.28	99.32
5	134	5.06	185.78
10	144	2.81	470.69
20	151	1.24	1044.05
40	146	0.36	2074.35
60	124	0.13	2955.51
80	107	0.06	3637.78
100	98	0.03	4229.89

Table 35: Numerical results: grid **E**

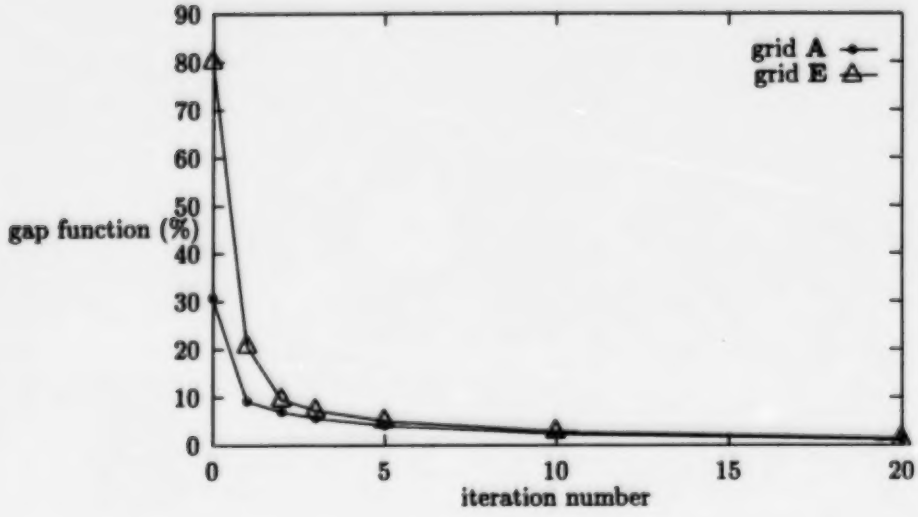

Figure 13: Decrease of the gap function for grids A and E (STRATEQ2)

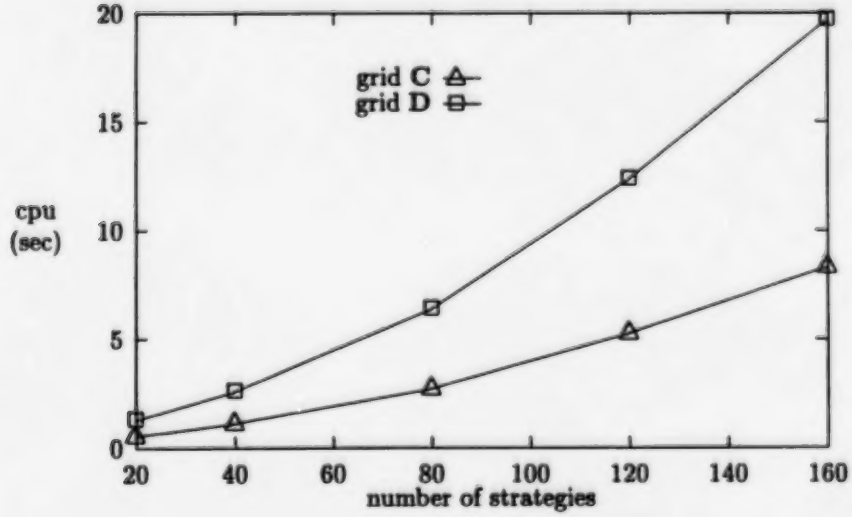

Figure 14: Running time per iteration for grids C and D (STRATEQ2)

	Cpu per iteration (sec)				
$\|W\|$	**A**	**B**	**C**	**D**	**E**
10	0.04	0.08	–	–	–
20	0.07	0.14	0.55	1.27	–
40	0.14	0.30	1.12	2.58	8.83
80	0.35	0.87	2.73	6.40	18.01
120	0.75	1.84	5.27	12.35	32.30
160	1.13	2.91	8.33	19.69	47.83

Table 36: Running times per iteration (**STRATEQ2**)

		Relative	
# cycles	$\|W\|$	gap (%)	Cpu (sec)
0	134	5.1×10^{0}	120
1	53	6.3×10^{-2}	241
2	50	5.0×10^{-2}	273
3	51	4.0×10^{-2}	306
4	51	5.0×10^{-2}	340
5	51	4.0×10^{-2}	376

Table 37: Projection algorithm applied to grid **E**

10 Conclusion

In this paper, we have shown that the concept of 'strategy', previously proposed as a paradigm for transit assignment models, can be adapted, in a nontrivial fashion, to the realm of traffic assignment on capacitated, acyclic networks. Although the model lacks the theoretical properties that would guarantee the uniqueness of the equilibrium solution, our algorithms converged to identical solutions on randomly generated networks, which is reassuring.

The next step of our study will be the analysis of a modified model involving access priorities at the tail nodes of capacitated arcs. Such a model would subsume a time-discretized formulation of a dynamic traffic assignment model where the FIFO rule would be automatically satisfied and where the capacity of the arcs would be an integral part of the model. This time-dependent network on which the model is based will naturally be topologically ordered with respect to the time increments, thus fulfilling the acyclicity condition required for the implementability of the loading procedure CAPLOAD.

References

[1] Vittorio Astarita, *A continuous time link model for dynamic network loading based on travel time function*, Proc. 13th International Symposium on Transportation and Traffic Theory (Tarrytown, N.Y.) (J.B. Lesort, ed.), Pergamon-Elsevier, 1996, pp. 79–102.

[2] D. Braess, *Über ein Paradoxen der Verkehrsplannung*, Unternehmenforschung **12** (1968), 258–268.

[3] Carlos F. Daganzo, *Queue spillovers in transportation networks with a route choice*, Transportation Science **32** (1998), 3-11 .

[4] M. Frank and P. Wolfe, *An algorithm for quadratic programming*, Naval Research Logistics Quarterly **3** (1956), 95–110.

[5] D. W. Hearn, *Bounding flows in traffic assignment models*, Research report 80-4, Department of Industrial and Systems Engineering, University of Florida, Gainsville, 1980.

[6] I. Konnov, *Combined relaxation methods for finding equilibrium points and solving related problems*, Mathematika **37** (1993), 44–51.

[7] G. M. Korpelevich, *The extragradient method for finding saddle points and other problems*, Matekon **13** (1977), 35–49.

[8] T. Larsson and M. Patriksson, *Side constrained traffic equilibrium models - traffic management through link tolls*, Equilibrium and Advanced Transportation Modelling (P. Marcotte and S. Nguyen, eds.), Kluwer, Massachusetts, 1998, pp. 125–151.

[9] S. Lawphongpanich and D. W. Hearn, *Simplicial decomposition of the asymmetric traffic assignment problem*, Transportation Research **18B** (1984), 123–133.

[10] P. Marcotte and S. Nguyen, *Hyperpath formulations of traffic assignment problems*, Equilibrium and Advanced Transportation Modelling (P. Marcotte and S. Nguyen, eds.), Kluwer Academic Publisher, 1998, pp. 175–199.

[11] S. Nguyen and S. Pallottino, *Equilibrium traffic assignment for large scale transit network*, European Journal of Operational Research **37** (1988), 176–186.

[12] S. Nguyen and S. Pallottino,*Hyperpaths and shortest hyperpaths*, Combinatorial Optimization (Berlin) (B. Simeone, ed.), Lecture Notes In Mathematics, vol. 1403, Springer-Verlag, 1989, pp. 258–271.

[13] Alexandre Schoeb, *Une étude théorique et algorithmique d'un modèle d'affectation de trafic avec capacités rigides*, mémoire de de maitrîse, Département d'informatique et de recherche opérationnelle, Université de Montréal, 1999.

[14] H. Spiess and M. Florian, *Optimal strategies: A new assignment model for transit networks*, Transportation Research B **23** (1989), no. 2, 83–102.

[15] C. Suwansirikul, T.L. Friesz, and R.L. Tobin, *Equilibrium decomposition optimization : a heuristic for the continuous equilibrium network design problem*, Transportation Science **21** (1987), 254–263.

[16] J. G. Wardrop, *Some theoretical aspects of road traffic research*, Proc. Inst. Civil Engr., Part 2, no. 1, 1952, pp. 325–378.